TECHNICAL MEMORANDUM NASA 44

AN ASSEMBLER FOR THE MOS TECHNOLOGY 6502 MICROPROCESSOR AS IMPLEMENTED IN JOLT (TM) AND KIM-1 (TM)

The 6502 Assembler implemented at Ohio University for support of microprocessor program development in the Tri-University Program is described.

by

Robert W. Lilley
Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

November 1976

Supported by

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia
Grant NGR 36-009-017

TABLE OF CONTENTS

I. INTRODUCTION

The development of computer software for microprocessors is materially aided by the assembler program; such programs generally allow the use of mnemonic variable names instead of absolute addressing and they will compute relative addresses for branching instructions and other useful functions. The programmer's task can then emphasize program content rather than the specific forms required by the target microprocessor instruction set.

The NASA Tri-University team at Ohio University is designing low-cost, microcomputer-based navigation receivers, and the assembler described in this paper was implemented for support of this project effort.

Ohio University provides computer services to its departments from a central site utilizing remote communication terminals. The flexibility of the environment provided by IBM's Virtual Machine Facility [1] and the Conversational Monitor System [2,3] make possible the convenient assembler access described herein.

This implementation of the assembler for the MOS Technology 6502 microprocessor chip serves a part of the present need; it forms a model for support of other microprocessors, for which we expect to have applications in the future.

The 6502 Assembler is in current use for development of Omega navigation software for the Ohio University Software-Based Receiver, the developmental models of which use the JOLT (TM) and KIM-1 (TM) microcomputer hardware.

II. MOS TECHNOLOGY 6502 MICROPROCESSOR INSTRUCTIONS

The MOS Technology 6502 Microprocessing Unit (MPU) integrated circuit chip is used in the JOLT and KIM-1 microcomputer units, in combination with appropriate read-only-memory and random-access memory chips. The MPU chip has some 55 unique operations, each of which may be performed upon data in a variety of ways. Some thirteen addressing modes allow flexibility in applying the 6502s basic logical operations to data.

Figure 1 reproduces the JOLT microcomputer reference data for addressing modes and instructions. The KIM-1 data are identical.

The complete descriptions of the 6502 addressing modes and instructions are contained in the JOLT and KIM-1 literature (see references 4,5,6); they will not be repeated here. The 6502 Assembler accepts the mnemonic forms of all 6502 instructions as operation codes. Addressing modes are determined by the type of assembler operand entered with the operation code. Assembler statements are described in Section III of this paper.

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A+M+C → A (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	–	–	✓
AND	A∧M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	–	–	–	–
ASL	C ← 7 ← 0 ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	–	–	–
BCC	BRANCH ON C=0 (2)																															90	2	2							–	–	–	–	–	–
BCS	BRANCH ON C=1 (2)																															B0	2	2							–	–	–	–	–	–
BEQ	BRANCH ON Z=1 (2)																															F0	2	2							–	–	–	–	–	–
BIT	A∧M				2C	4	3	24	3	2																															M_7	✓	–	–	–	M_6
BMI	BRANCH ON N=1 (2)																															30	2	2							–	–	–	–	–	–
BNE	BRANCH ON Z=0 (2)																															D0	2	2							–	–	–	–	–	–
BPL	BRANCH ON N=0 (2)																															10	2	2							–	–	–	–	–	–
BRK	See Fig. 1													00	7	1																									–	–	–	1	–	–
BVC	BRANCH ON V=0 (2)																															50	2	2							–	–	–	–	–	–
BVS	BRANCH ON V=1 (2)																															70	2	2							–	–	–	–	–	–
CLC	0 → C													18	2	1																									–	–	0	–	–	–
CLD	0 → D													D8	2	1																									–	–	–	–	0	–
CLI	0 → I													58	2	1																									–	–	–	0	–	–
CLV	0 → V													B8	2	1																									–	–	–	–	–	0
CMP	A−M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	–	–	–
CPX	X−M	E0	2	2	EC	4	3	E4	3	2																															✓	✓	✓	–	–	–
CPY	Y−M	C0	2	2	CC	4	3	C4	3	2																															✓	✓	✓	–	–	–
DEC	M−1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	–	–	–	–
DEX	X−1 → X													CA	2	1																									✓	✓	–	–	–	–
DEY	Y−1 → Y													88	2	1																									✓	✓	–	–	–	–
EOR	A ⊻ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	–	–	–	–
INC	M+1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	–	–	–	–
INX	X+1 → X													E8	2	1																									✓	✓	–	–	–	–
INY	Y+1 → Y													C8	2	1																									✓	✓	–	–	–	–
JMP	JUMP TO NEW LOC.				4C	3	3																												6C	5	3				–	–	–	–	–	–
JSR	(See Fig. 2) JUMP SUB				20	6	3																																		–	–	–	–	–	–
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	–	–	–	–

MNEMONIC	OPERATION	IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3										✓	✓	–	–	–	–
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													✓	✓	–	–	–	–
LSR	0 → 7 → 0 → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0	✓	✓	–	–	–
NOP	NO OPERATION													EA	2	1																									–	–	–	–	–	–
ORA	A∨M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓	✓	–	–	–	–
PHA	A → M_S S−1 → S													48	3	1																									–	–	–	–	–	–
PHP	P → M_S S−1 → S													08	3	1																									–	–	–	–	–	–
PLA	S+1 → S M_S → A													68	4	1																									–	–	–	–	–	–
PLP	S+1 → S M_S → P													28	4	1																									(RESTORED)					
ROL	C 7 ← 0 ← 0				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													✓	✓	✓	–	–	–
RTI	(See Fig. 1) RTRN. INT.													40	6	1																									(RESTORED)					
RTS	(See Fig. 2) RTRN SUB													60	6	1																									–	–	–	–	–	–
SBC	A−M−$\bar{C}$ → A (1) (3)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										✓	✓	(3)	–	–	✓
SEC	1 → C													38	2	1																									–	–	1	–	–	–
SED	1 → D													F8	2	1																									–	–	–	–	1	–
SEI	1 → I													78	2	1																									–	–	–	1	–	–
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										–	–	–	–	–	–
STX	X → M				8E	4	3	86	3	2																													96	4	2	–	–	–	–	–	–
STY	Y → M				8C	4	3	84	3	2													94	4	2																–	–	–	–	–	–
TAX	A → X													AA	2	1																									✓	✓	–	–	–	–
TAY	A → Y													A8	2	1																									✓	✓	–	–	–	–
TSX	S → X													BA	2	1																									✓	✓	–	–	–	–
TXA	X → A													8A	2	1																									✓	✓	–	–	–	–
TXS	X → S													9A	2	1																									–	–	–	–	–	–
TYA	Y → A													98	2	1																									✓	✓	–	–	–	–

(1) ADD 1 TO "N" IF PAGE BOUNDRY IS CROSSED.
(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE.
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE.
(3) CARRY NOT = BORROW.

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
M_S MEMORY PER STACK POINTER

+ ADD
− SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
✓ MODIFIED
– NOT MODIFIED
M_7 MEMORY BIT 7
M_6 MEMORY BIT 6

OP-CODE TABLE

LSD / MSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	LSD / MSD
0	BRK	ORA IND,X				ORA Z.Page	ASL Z.PAGE		PHP	ORA IMM	ASL A			ORA ABS	ASL ABS		0
1	BPL	ORA IND,Y				ORA Z.Page,X	ASL Z.Page,X		CLC	ORA ABS,Y				ORA ABS,X	ASL ABS,X		1
2	JSR	AND IND,X			BIT Z.Page	AND Z.Page	ROL Z.Page		PLP	AND IMM	ROL A		BIT ABS	AND ABS	ROL ABS		2
3	BMI	AND IND,Y				AND Z.Page,X	ROL Z.Page,X		SEC	AND ABS,Y				AND ABS,X	ROL ABS,X		3
4	RTI	EOR IND,X				EOR Z.Page	LSR Z.Page		PHA	EOR IMM	LSR A		JMP ABS	EOR ABS	LSR ABS		4
5	BVC	EOR IND,Y				EOR Z.Page,X	LSR Z.Page,X		CLI	EOR ABS,Y				EOR ABS,X	LSR ABS,X		5
6	RTS	ADC IND,X				ADC Z.Page			PLA	ADC IMM			JMP IND	ADC ABS			6
7	BVS	ADC IND,Y				ADC Z.Page,X			SEI	ADC ABS,Y				ADC ABS,X			7
8		STA IND,X			STY Z.Page	STA Z.Page	STX Z.Page		DEY		TXA		STY ABS	STA ABS	STX ABS		8
9	BCC	STA IND,Y			STY Z.Page,X	STA Z.Page,X	STX Z.Page,Y		TYA	STA ABS,Y	TXS			STA ABS,X			9
A	LDY IMM	LDA IND,X	LDX IMM		LDY Z.Page	LDA Z.Page	LDX Z.Page		TAY	LDA IMM	TAX		LDY ABS	LDA ABS	LDX ABS		A
B	BCS	LDA IND,Y			LDY Z.Page,X	LDA Z.Page,X	LDX Z.Page,Y		CLV	LDA ABS,Y	TSX		LDY ABS,X	LDA ABS,X	LDX ABS,Y		B
C	CPY IMM	CMP IND,X			CPY Z.Page	CMP Z.Page	DEC Z.Page		INY	CMP IMM	DEX		CPY ABS	CMP ABS	DEC ABS		C
D	BNE	CMP IND,Y				CMP Z.Page,X	DEC Z.Page,X		CLD	CMP ABS,Y				CMP ABS,X	DEC ABS,X		D
E	CPX IMM	SBC IND,X			CPX Z.Page	SBC Z.Page	INC Z.Page		INX	SBC IMM	NOP		CPX ABS	SBC ABS	INC ABS		E
F	BEQ	SBC IND,Y				SBC Z.Page,X	INC Z.Page,X		SED	SBC ABS,Y				SBC ABS,X	INC ABS,X		F

IMM IMMEDIATE ADDRESSING – THE OPERAND IS CONTAINED IN THE SECOND BYTE OF THE INSTRUCTION.

ABS ABSOLUTE ADDRESSING – THE SECOND BYTE OF THE INSTRUCTION CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE THIRD BYTE CONTAINS THE 8 HIGH ORDER BITS OF THE EFFECTIVE ADDRESS.

Z PAGE ZERO PAGE ADDRESSING – SECOND BYTE CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE 8 HIGH ORDER BITS ARE ZERO.

A ACCUMULATOR – ONE BYTE INSTRUCTION OPERATING ON THE ACCUMULATOR.

Z PAGE X, Z PAGE Y, ZERO PAGE INDEXED – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE INDEX (CARRY IS DROPPED) TO FORM THE LOW ORDER BYTE OF THE EA. THE HIGH ORDER BYTE OF THE EA IS ZEROS.

ABS X, ABS Y ABSOLUTE INDEXED – THE EFFECTIVE ADDRESS IS FORMED BY ADDING THE INDEX TO THE SECOND AND THIRD BYTE OF THE INSTRUCTION.

IND,X INDEXED INDIRECT – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE X INDEX, DISCARDING THE CARRY. THE RESULTS POINTS TO A LOCATION ON PAGE ZERO WHICH CONTAINS THE 8 LOW ORDER BITS OF THE EA. THE NEXT BYTE CONTAINS THE 8 HIGH ORDER BITS.

(IND),Y INDIRECT INDEXED – THE SECOND BYTE OF THE INSTRUCTION POINTS TO A LOCATION IN PAGE ZERO. THE CONTENTS OF THIS MEMORY LOCATION IS ADDED TO THE Y INDEX. THE RESULT BEING THE LOW ORDER EIGHT BITS OF THE EA. THE CARRY FROM THIS OPERATION IS ADDED TO THE CONTENTS OF THE NEXT PAGE ZERO LOCATION, THE RESULT BEING THE 8 HIGH ORDER BITS OF THE EA.

Figure 1. JOLT or KIM-1 Operation Codes and Addressing Modes Summary.

III. 6502 ASSEMBLER LANGUAGE

This description of the 6502 Assembler Language provides a user's guide to writing assembler statements for the MOS Technology 6502 microcomputer chip instruction set. The assembler allows selection of basic 6502 operation codes (opcode) or certain additional operations defined by the assembler which aid the programmer in establishing data areas, constants and program flow control. This description is heavily dependent on Rankin's Cross-Assembler Manual. [7]

Assembler input consists of a group of assembler statements followed by an END assembler instruction. Outputs consist of a "clean" (formatted) version of the input statements, error messages as appropriate, and a hexidecimal output file containing program object code in a format readable by JOLT or KIM-1 hex program loaders.

A. Assembler Source Statement Format. A source line has four parts: a label field, an opcode field, an operand field, and a comment field. The fields are defined by one to any number of spaces separating them. A total line is up to 72 characters long.

Label - A label must start in the first position in the line. If the first position is blank, no label is assumed. A label must start with an alphabetic character and can be 1 to 6 characters in length. All base page labels must appear before their first use as an operand.

" * " in the first position of the line indicates that the line is to be taken as a comment.

Opcode - The opcode is three characters long always preceeded by at least one space. Opcodes are of two forms. The first form causes the creation of a machine instruction (1 to 3 bytes) and is known as a machine operation code (opcode). The second form is either a control statement to the assembler, or it defines constants, addresses, or symbols and is known as pseudo-opcode. Each pseudo-opcode will be explained below.

Operand - Most, although not all, opcodes require an operand for additional information, such as an address. An operand is preceeded by, and terminated by, a space. The exact form of the operand determines in part which addressing mode is to be used.

Comments - After the operand (or opcode if no operand required) and a trailing space, the remainder of the line can be devoted to comments and is not processed by the assembler.

1. Addressing Modes. The 6502 has 13 addressing modes. Consult the summary sheet, Figure 1, for which instructions use which modes, and for mnemonics. The following paragraphs give addressing modes and operands required.

Implied - Any instruction which uses implied addressing does not need an operand.

Immediate - This mode places the operand value as the second byte of the assembled instruction. The operand has the form =Label [+/- #], = # , or =' &'. Where [...] is optional,

and +/- means either + or - . # is a number which has the following prefixes:

None	Decimal
&	Octal
$	Hexadeimal
%	Binary

Relative - Relative addressing is used for branching statements and has two forms. In the first form, the operand is label [+/- #] or # . In this form the operand must be within +127 to -128 of the current program counter (when pointing to the start of the next instruction). In the second form the offset is given by +/- # where +/- # must be in the range +127 to -128 and is the relative jump from the start of the next instruction.

Absolute - Absolute addressing creates 3 bytes of machine code (1 for the instruction and 2 for address). It has the form label [+/- #], #, or * [+/- #], where * is the value of the program counter.

Z. Page - Zero-page addressing is same as absolute except that the address has a value of less than 256. Note that labels appearing in the operand must be previously defined or improper assembly will result.

Accumulator - Operand is A. A is a reserved symbol and cannot be used otherwise as a label.

Indexed - Indexed addressing can either be zero-page or absolute and use either index register (if allowed by the instruction). The form is Add, X or Add, Y where Add is an address formed as described for addresses in absolute and zero page addressing.

Indexed Indirect - Indexed indirect addressing uses only the X register. The operand is added to X which then points to a full address stored in the base page. Form is (Add, X) where Add is an address as described in zero page addressing.

Indirect Indexed - Y is added to the address in Add with carry added to address in Add+1. The fetch is obtained from the resulting memory location. Form is (Add), Y where Add has same form as above.

Indirect - Use only in the JMP instruction and has form (Add).

B. Assembler Instructions (Pseudo-Opcodes).

ORG #	Sets program counter to #
END	Ends Assembly

Label EQU M - Assigns the value of M to label rather than the value of the program counter. M may contain a simple expression consisting of Label [+/-#], or #. M must be positive and label previously defined. No code is generated.

[Label] ADR Lab [+/-#] - Places value of operand into memory. If both operand and PC are less than 256,1 byte of code is created, otherwise 2 bytes. Normally the operand is an address.

[Label] ASC@@@@ -Stores up to 40 ASCII characters in memory. 1 byte of code is generated for each character.

[Label] OCT Num [,Num...] - Stores in memory up to 40 octal numbers where Num is 1 to 40 Octal numbers separated by commas. Each number results in 2 bytes of code. If number has prefix of - , the 2 S complement of the number is stored. Base prefixes must not be used in Num.

[Label] HEX Num [,Num...] - Same as OCT except that Num is in HEX and 1 byte is generated for each number.

[Label] DCM Num [,Num...] - Same as OCT except Num is in decimal.

[Label] INT Num [,Num...] - Same as DCM except 1 byte is generated for each number.

[Label] BCD Num [,Num...] - Same as INT except numbers are stored in BCD and negative numbers stored as 10's complement.

C. Example Statements.

```
 1                *
 2                *          This program is to demonstrate addressing modes
 3                *          and does not represent an actual program
 4   0010                    ORG $10
 5   0010         Label1 BSS 10
 6   001A 10              ADR Label1
 7   0100                    ORG 256
 8   0100         J      EQU 10
 9   0100 31      Label 2 ASC '123'
     0101 32
     0102 33
10   0103 00              OCT 177400,-1
          FF
     0105 FF
          FF
11   0107 00              HEX 00,-00,AA
     0108 00
     0109 AA
12   010A 28              DCM 9000
          23
13   010C 10              INT 16
14   010D 99              BCD 99,00,01
     010E 00
     010F 0i
```

```
14  0110 EA          NOP              Implied Addressing
15  0111 DO          BNE Label 3      Relative Addressing
           03
16  0113 AD          LDA Label 2      Absolute Addressing
           00
           01
17  0116 AE  Label 3 LDX $105           "
           05
           01
18  0119 25          AND Label 1      Z. Page ADD.
           10
19  011B 4A          LSR A            Accumulator ADD.
20  011C EO          CPX = J          Immediate ADD.
           OA
21  011E D5          CMP Label 1+5,X  Indexed ADD.
           15
22  0102 41          EOR (Label 1,X)  Indexed Indirec.
           10
23  0122 F1          SBC (Label 1-1),Y  Indirect Indexed
           OF
24  0124 6C          JMP (Label 2)    Indirect
           01
25                   END
```

IV. USE OF THE 6502 ASSEMBLER IN THE CMS ENVIRONMENT

At Ohio University, central computer resources are available through use of remote terminals connected to an IBM System/370 Model 158 computer system running the Virtual Machine Facility (VM/370). The 6502 Assembler is stored on the Conversational Monitor System (CMS) virtual disk assigned to virtual machine AVENCTR. This disk is also available to virtual machine AVIONICS in read-only mode. In practice, AVENCTR access is used for assembler maintenance and disk storage. AVIONICS machine access is generally used for operation of the assembler to produce hex input tapes for the JOLT or KIM-1 microcomputers used in Omega navigation work.

The remainder of this section is devoted to describing the CMS environment in which the user accesses the 6502 assembler, and it presumes a working knowledge of CMS on the part of the reader. Assembler maintenance is discussed in Section V.

A. Assembler Operation. First, the user must establish communication with the VM/370 system. Either by dialing the telephone number for 110-baud or 300-baud operation, as appropriate for the terminal device, or by powering-up the terminal for direct-connected devices, the user will receive the "VM/370 ONLINE" message. Type a carriage return (CR) in reply. After the prompting dot, type LOGON AVIONICS (CR). The system will reply ENTER PASSWORD and type a mask to avoid unauthorized observation of the password. After the mask is typed, enter the password in current use for machine AVIONICS. The

system replies with the logon message, and ends with a prompting dot. The user is now ready to begin CMS programming.

The 6502 Assembler is capable of operation in various modes, with various requirements from the user. In all cases, the assembler requires source program input either from the terminal or from a CMS disk file. Terminal input is entered as user responses, line-by-line, to prompting dots produced by the assembler. CMS disk files are produced by use of the CMS Editor. The EDIT command begins the file-building process.

CMS files have compound names consisting of three parts. The user must enter all three parts to describe completely the file. For 6502 Assembler operation, the first two parts of the file description are arbitrary. The user may name his file anything he wishes as long as the name does not conflict with an existing file name in his CMS library. (The CMS "LISTFILE" command may be used to print library file names.) The third part of the file description is, for our purposes, either "A" or "C". This character tells which CMS "disk" will be used for file storage.

As the AVIONICS and AVENCTR machines are currently configured, the A-disk is permanent storage for CMS files. The disk will remain, even though the user logs off the machine, to return later. Upon LOGON, a C-disk is formed, as temporary storage for CMS file use in the current terminal session. This implies that additional storage space is available to the user for temporary files of his own, or for work files produced by the 6502 Assembler. The user can put files on the C-disk by using the character "C" as the third part of his file description, or by copying files from the A-disk. The 6502 Assembler will use the C-disk for all its work files to avoid crowding the A-disk (permanent file space) with temporary files which will no longer be needed after assembly is complete. For reasons which will become clear later, it is suggested that the user begin file construction on the C-disk, copying correct files to the A-disk when desired for permanent storage.

B. Suggested Usage of the Assembler and Files. After LOGON, the user has a choice of using the Assembler to assemble input files directly from the terminal, or from CMS input files. To assemble from the terminal, the user begins the process by entering

JASM TERMINAL (See Footnote 1)

in response to a prompting dot from CMS. This command invokes the 6502 Assembler and replies "EXECUTION BEGINS..." and then issues a prompting dot. At this point, the user may type 6502 Assembler statements directly to Pass 1 of the Assembler. When the Assembler "END" statement is entered, the Assembler enters Pass 2 and produces an output listing on the terminal inciuding any error messages necessary. When Pass 2 is completed, the following files will be found on the C-disk:

---

1 The JASM command, and others to be described later, is a CMS EXEC procedure, written for this application. CMS allows these pseudo-commands to be built and executed by the user. See Appendix B for EXEC listings.

JOLT CLEAN C - A listing of the user's input code, arranged in columnar form for ease in reading. This file could be copied to the A-disk (or to another C-disk file name) for retention and subsequent updating, if desired. In fact, it must be copied or renamed before another JASM command is issued or it will be replaced by another JOLT CLEAN C file from the latest assembly. If the file is renamed, even though it is left on C-disk, the new JOLT CLEAN C file from the next assembly will not replace it.

JOLT HEXCODE C - A file containing the hex "object code" in hextape format for reading by JOLT of KIM-1. To obtain this file on paper tape for input to the microcomputer, the user must output the file to the Model 33 Teletype (TM) unit using the CMS "TYPE" command. If the user is operating on another terminal type which has no paper tape capability, he must save the JOLT HEXCODE C file by copying it to the A-disk, logging off, and logging on later using the Model 33 terminal and issuing the TYPE command for the file name he assigned the file when he copied it.

For example:

```
LOGON AVIONICS
JASM TERMINAL
(entry of assembler statements)
COPY JOLT  CLEAN C TEST PROG A
COPY JOLT HEXCODE C TEST HEX A
LOGOFF
```

Later, on the Model 33:

```
LOGON AVIONICS
TYPE TEST HEX A     (with tape punch on before CR).
```

These commands produce the assembled hexcode file, which the user saved on the A-disk as TEST HEX A, and the cleaned-up input file, which the user saved as TEST PROG A before logoff. Later, when the Model 33 terminal with tape punch became available, the user logged on and typed the hex file with the punch turned on to produce a tape for entry to the JOLT of KIM-1.

In many cases, this JASM TERMINAL mode may serve the user's need; one disadvantage to this method exists, however. Note that input from the terminal goes directly to the Assembler. Therefore, input errors may not be corrected by the user before the assembly proceeds. The CMS system allows another operational mode which circumvents this disadvantage. By building a CMS file of Assembler input statements using the CMS Editor, the user can take advantage of this very powerful edit capability to correct known errors in his input prior to assembly. After LOGON, the sequence is as follows:

EDIT MYFILE TEST A

or EDIT MYFILE TEST C

Where MYFILE and TEST are arbitrary names assigned to the file by the user, and the A or C denotes on which CMS disk the file will reside. It matters little whether the user specifies A or C at this point; remember, however, that any files to be retained after LOGOFF must be on the A-disk. The user then enters CMS input mode by issuing the INPUT command and begins typing Assember input statements according to the formats given earlier. In INPUT model, all typed input is stored, line-by-line, in the file MYFILE TEST. If an error is detected, the user may elect to finish typing input and then go back to correct it. If he desired to correct is when it occurred, he could enter EDIT mode with a null CR and use CMS edit commands to change the erroneous line, then re-enter INPUT mode to continue building the file.

When the file is complete, the user must enter EDIT mode with the null CR and store the file on disk using the CMS FILE command.

(Note: Use of the CMS AUTOSAVE command can aid in minimizing loss of data in case of a machine malfunction. See the CMS Command Guide.)

The user now has a 6502 Assembler input file stored on disk ready for assembly. He should now issue:

JASM MYFILE TEST A or JASM MYFILE TEST C

to retrieve the file and invoke the Assembler. From this point on, assembly proceeds as above, with JOLT HEXCODE C and JOLT CLEAN C files built. A program listing is typed at the terminal with error messages as appropriate. The file copying considerations are the same as above, except that the user may want to replace his input file with the JOLT CLEAN C file to take advantage of the neat formatting performed by the Assembler, facilitating later file update or correction. To do this, the user must issue:

COPY JOLT CLEAN C MYFILE TEST A (REPLACE)

This command takes the JOLT CLEAN FILE and replaces the user file (here assumed to be on disk A) with it. The user may also want to copy the JOLT HEXCODE C file for later use, if it is error-free and valuable.

One other option is allowed for assemblies from CMS input files. If the user is working with a large assembly, the time taken to type the listing on the terminal for each assembly may be prohibitive. Instead, the user may wish to keep a "master" listing on paper and update it by hand as program development proceeds. Then he can obtain a program listing only after a series of changes to the input file, or after a major change which renders his paper listing obsolete. To obtain an assembly without terminal listing he should issue:

JASM MYFILE TEST A NOLIST

Where, once again, MYFILE and TEST are arbitrary names assigned by the user, and A denotes on which CMS disk the file is residing. The NOLIST option prevents the printing of any terminal output during assembly. Instead, the listing goes into file JOLT PRINTOUT C for possible later reference.

The user may well want to scan this disk-resident printout file for errors after each NOLIST assembly. To do this, issue:

JERRS

The JERRS command causes a scan of the JOLT PRINTOUT C file and prints any error messages and the 6502 Assembler lines which generated them for the user's review. Note that JOLT PRINTOUT C is only produced when assembly is done using the NOLIST option on the JASM command. If JERRS is issued after a normal assembly, no PRINTOUT file will be found and no output of errors will be given.

The user may find it useful to refer to the flow chart of Figure 2 and to the sample terminal session of Figure 3 for additional information.

V. ASSEMBLER MAINTENANCE SUPPORT UNDER CMS

Maintenance of the 6502 Assembler generally is done using the AVENCTR virtual machine. The master copy of the Assembler is located on the A-disk as file ASM6502 FORTRAN AO, making it private to this machine. The file contains the FORTRAN source statements for the assembler (See Appendix D). A second copy is stored as ASM6502 PFORTRAN AO, in a packed format for backup in case of loss of the primary copy.

Maintenance usually takes the form of some update or alteration of the source code and then a series of tests to verify that the resulting Assembler operates properly. For this purpose, a set of maintenance EXEC procedures are provided in the AVENCTR A-disk library. MBUILD provides for compilation of the ASM6502 FORTRAN file into a MAINT MODULE C file which is then executed for testing. Later use of the MAINT MODULE C file is made by MASM, which works as does JASM, described earlier, except that the MAINT MODULE file is used instead of the operational JOLT MODULE A file.

Correct forms for these maintenance EXECs are:

```
    MBUILD TERMINAL
or  MBUILD ASMBL TEST C
or  MBUILD ASMBL TEST C NOLIST
    MASM TERMINAL
or  MASM TEST FILE C
or  MASM TEST FILE C NOLIST
```

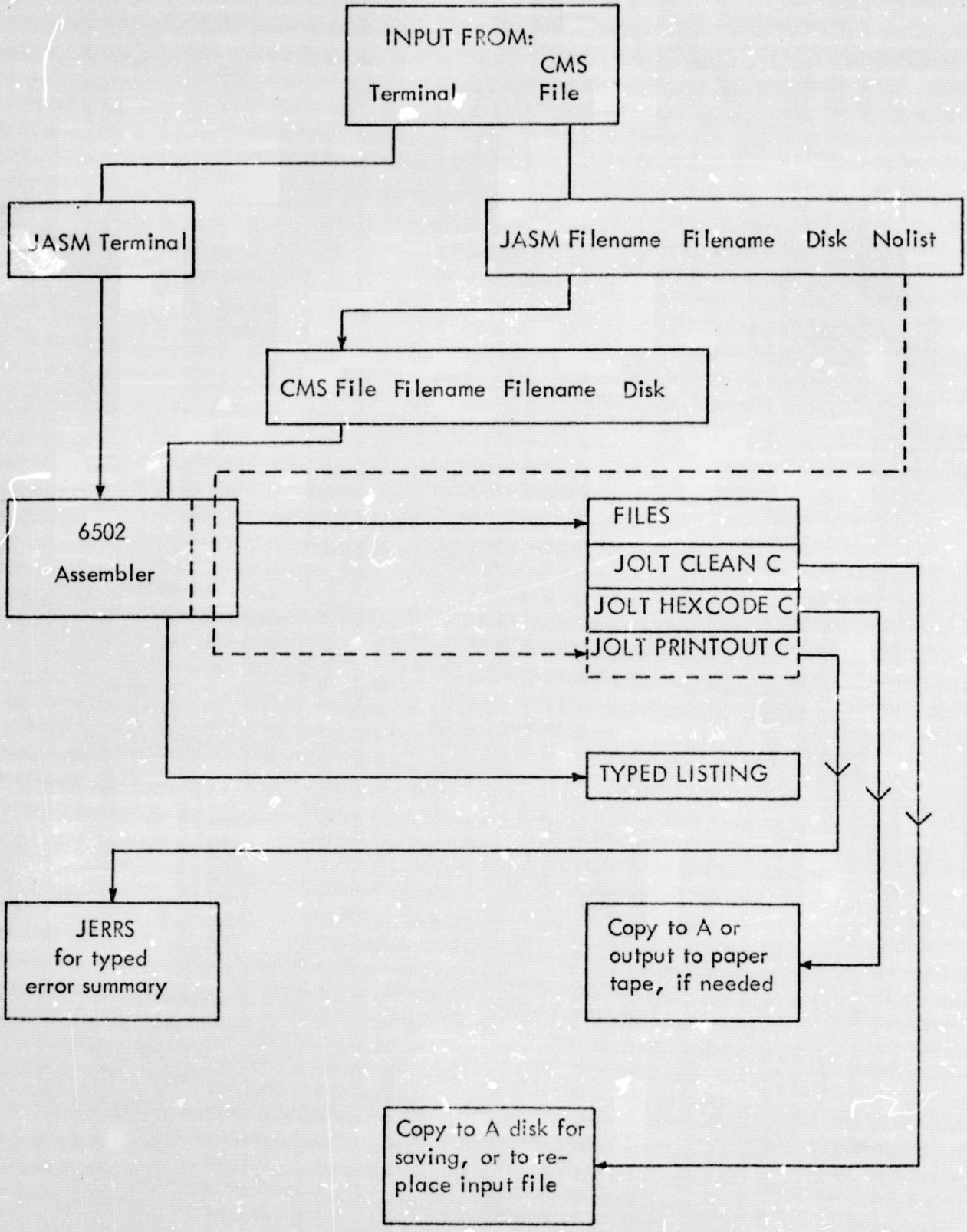

Figure 2. 6502 Assembler Data Flow.

```
VM/370 ONLINE
!

.logon avionics
ENTER PASSWORD:
.********
LOGON AT 13:47:00 EDT FRIDAY 11/05/76
CMS VERSION 3.0 - 10/13/76 06:36
.edit jolt test c
D (192) R/O
READY: 1-CYL C-DISK ONLINE
NEW FILE:
EDIT:
.input
INPUT:
. org $100
. lda label1
. sta label2
. brk
.label1 bss 1
.label2 bss 1
. end
.
EDIT:
.file
R;

.jasm jolt test c
EXECUTION BEGINS...

END PASS 1:    0 ERRORS

   1  0100              ORG $100
   2  0100 AD           LDA LABEL1
           07
           01
   3  0103 8D           STA LABEL2
           08
           01
   4  0106 00           BRK
   5  0107 00  LABEL1   BSS 1
   6  0108 00  LABEL2   BSS 1
   7                    END

END PASS 2:    0 ERRORS

R;

.type jolt hexcode c

;0901000AD07018D0801000000000155
;00

R;

.type jolt clean c
```


Figure 3. Sample Terminal Session.

```
        ORG $100
        LDA LABEL1
        STA LABEL2
        BRK
LABEL1  BSS 1
LABEL2  BSS 1
        END

R;

.copy jolt clean c jolt prog1 a
R;

.logoff
CONNECT= 00:09:09 VIRTCPU= 000:00.58 TOTCPU= 000:02.30
LOGOFF AT 13:56:09 EDT FRIDAY 11/05/76
```


Figure 3. Sample Terminal Session (Cont.).

Again, file names are arbitrary, and the same files are built as described earlier for hex codes and clean listings. To avoid interference with current operating modules, however, all these maintenance files have MAINT as the first part of the file name.

When maintenance and testing are complete, the operative MAINT MODULE C must be copied to replace the current version of JOLT MODULE C on the AVENCTR A-disk. Then, subsequent use of JASM on either AVIONICS or AVENCTR machines will result in assemblies using the updated Assembler.

VI. ACKNOWLEDGMENTS

The author acknowledges the origin of the 6502 Assembler with Roy R. Rankin at Stanford University, and the assistance of Mr. Lynn Smith at Microcomputer Associates, Inc., who provided a copy of this program, which appears in altered form in this paper.

Richard Salter, of the Ohio University Omega Team, has been the principal Assembler user and offered many suggestions for improvements in operation. Ralph W. Burhans is the Project Engineer for Ohio University's portion of the NASA Joint University Program, and Dr. Richard H. McFarland is Director of the Avionics Engineering Center. Their contributions are appreciated.

VII. REFERENCES

[1] IBM Virtual Machine Facility/370; CP Command Reference for General Users, File No. 5370-36; GC20-1820-0.

[2] IBM Virtual Machine Facility/370; CMS Command and Macro References, File No. 5370-36; GC20-1818-0.

[3] IBM Virtual Machine Facility/370; CMS User's Guide, File No. 5370-30; GC20-1819-0.

[4] MCS6500 Microcomputer Family Programming Manual, MOS Technology, Inc., Norristown, Pennsylvania, January, 1976.

[5] JOLT Demon Software Manual, Microcomputer Associates, Inc., Santa Clara, California, 1975.

[6] KIM-1 User Manual, MOS Technology, Inc., Norristown, Pennsylvania, March, 1976.

[7] 6502 Cross-Assembler Manual, R. Rankin (unpublished) November, 1975.

VIII. APPENDICES

A. Assembler Source Listing.

B. CMS Exec Procedure Listings.

1. PROFILE
2. JASM
3. JERRS
4. MASM
5. MBUILD

APPENDIX A

```
      IMPLICIT INTEGER*2 (A-Z)                                          ASM00010
C     CROSS-ASSEMBLER FOR 6502 MICROPROCESSOR (REV. 1.0)                ASM00020
C                                                                       ASM00030
C     ROY R RANKIN                                                      ASM00040
C     STANFORD UNIVERSITY                                               ASM00050
C     415-497-1822                                                      ASM00060
C     DEC. 8, 1975                                                      ASM00070
C                                                                       ASM00080
C     SYSTEM/370 MODIFICATIONS - R. W. LILLEY, OHIO UNIV. AVIONICS      ASM00090
C     AUGUST, 1976                                                      ASM00100
C                                                                       ASM00110
C     THIS PROGRAM READS AN INPUT FILE CONTAINING THE ASSEMBLER CODE    ASM00120
C      (LOGICAL UNIT 5, LUIN) ONCE FOR EACH OF THE                      ASM00130
C     PROGRAM'S TWO PASSES. AN ASSEMBLY LISTING IS OUTPUT TO THE LINE   ASM00140
C     PRINTER (LOGICAL UNIT 6, LU) ALONG WITH AN ABSOLUTE FILE,         ASM00150
C     READABLE BY JOLT'S DEMON OR MOS TECHNOLOGY'S TIM, TO THE PUNCH    ASM00160
C     FILE (LOGICAL UNIT 4, LUP).                                       ASM00170
C                                                                       ASM00180
C     FILE 11 IS A SEQUENTIAL SCRATCH FILE FOR STORING INPUT TEXT       ASM00190
C     FOR PASS 2                                                        ASM00200
C                                                                       ASM00210
C     THE SYMBOL TABLE IS SET TO ALLOW 300 ENTRIES. TO ALLOW MORE ENTRIES ASM00220
C     SET THE DIMENSION OF ISYMB(3,N) AND SYMB(N) TO THE SIZE OF THE    ASM00230
C     DESIRED SYMBOL TABLE LENGTH AND SET NSYTM TO THE NUMBER OF        ASM00240
C     ENTRIES. THIS NEED BE DONE ONLY IN THE MAIN PROGRAM.              ASM00250
C                                                                       ASM00260
      DIMENSION IN(40),LABEL(3),EQU(2),IADD(2),IPC(2),OBUF(40)          ASM00270
      INTEGER LUP,LU,LUIN                                               ASM00280
      REAL PC,CODE(3)                                                   ASM00290
      DIMENSION POUT(30)                                                ASM00300
      DIMENSION ISYMB(3,300),SYMB(300)                                  ASM00310
      DATA STAR/'*'/,BLANK/' '/,LU/6/,LUIN/5/,LUP/4/,NSYTM/300/,NSYT/0/ ASM00320
      DATA MNMAX/26/                                                    ASM00330
      INTEGER OSK                                                       ASM00340
      REAL MONE                                                         ASM00350
      ZRO=0                                                             ASM00360
      ONE=1                                                             ASM00370
      MONE=-1.                                                          ASM00380
      AITY=80                                                           ASM00390
      TWO=2                                                             ASM00400
      OSW=0                                                             ASM00410
      OSK=11                                                            ASM00420
      MN=0                                                              ASM00430
      IPASS=1                                                           ASM00440
      PASS=1                                                            ASM00450
      NERR=0                                                            ASM00460
      PC=0                                                              ASM00470
      LINE=0                                                            ASM00480
   10 IF(LINE.LT.0)LINE=-LINE                                           ASM00490
      IF(OSW.EQ.1)GO TO 701                                             ASM00500
      READ(LUIN,1)IN                                                    ASM00510
      CALL LENT(IN)                                                     ASM00520
      WRITE(OSK,1)IN                                                    ASM00530
      GO TO 702                                                         ASM00540
  701 READ(OSK,1)IN                                                     ASM00550
```

```
  702 IERROR=0                                                          ASM00560
      LINE=LINE+1                                                       ASM00570
    1 FORMAT(40A2)                                                      ASM00580
      IBUFP=0                                                           ASM00590
      ICHAR=MASK2(SRS(IN(1)))                                           ASM00600
C                                                                       ASM00610
C   COMMENT?                                                            ASM00620
C                                                                       ASM00630
      IF(ICHAR.NE.STAR)GO TO 12                                         ASM00640
      IF(PASS.EQ.2)WRITE(LU,2)LINE,IN                                   ASM00650
      GO TO 10                                                          ASM00660
    2 FORMAT(1X,I4,11X,40A2)                                            ASM00670
C                                                                       ASM00680
C   LABEL?                                                              ASM00690
C                                                                       ASM00700
   12 IF(ICHAR.EQ.BLANK)GO TO 30                                        ASM00710
C                                                                       ASM00720
C   PROCESS LABEL                                                       ASM00730
C                                                                       ASM00740
      LABEL(1)=BLANK                                                    ASM00750
      LABEL(2)=BLANK                                                    ASM00760
      LABEL(3)=BLANK                                                    ASM00770
      IPBL=6                                                            ASM00780
      CALL GWORD(IN,IBUFP,AITY,LABEL,IPBL,IFLAG,ZRO)                    ASM00790
      IF(PASS.EQ.2)GO TO 30                                             ASM00800
      IF(IFLAG.NE.-1)GO TO 20                                           ASM00810
      WRITE(LU,8)LINE,LABEL                                             ASM00820
      NERR=NERR+1                                                       ASM00830
    8 FORMAT(1X,I4,' *** LABEL TOO LONG; USE ',3A2,' *** ')             ASM00840
C                                                                       ASM00850
C   PUT LABEL INTO SYMBOL TABLE                                         ASM00860
C                                                                       ASM00870
   20 CALL SYMBT(LABEL,PC,ISYMB,NSYTM,LU,NSYT,SYMB,LINE,NERR)           ASM00880
C                                                                       ASM00890
C   GET OPCODE                                                          ASM00900
C                                                                       ASM00910
   30 CALL OPCD(IN,IBUFP,BYTE,CODE,PASS,NSYT,ISYMB,SYMB,PC,IERROR,      ASM00920
     * OBUF,JK)                                                        ASM00930
C                                                                       ASM00940
C   ERROR PRINT                                                         ASM00950
C                                                                       ASM00960
      IF(IERROR.EQ.0.OR.PASS.EQ.1)GO TO 35                              ASM00970
      NERR=NERR+1                                                       ASM00980
      LINE=-LINE                                                        ASM00990
      GO TO(201,202,203,204,205,206,207,208,209,210,211),IERROR         ASM01000
  201 WRITE(LU,301)LINE                                                 ASM01010
  301 FORMAT(1X,I4,' *** LABEL UNDEFINED ***')                          ASM01020
      GO TO 35                                                          ASM01030
  202 WRITE(LU,302)LINE                                                 ASM01040
  302 FORMAT(1X,I4,' *** ILLEGAL OPERAND ***')                          ASM01050
      GO TO 35                                                          ASM01060
  203 WRITE(LU,303)LINE                                                 ASM01070
  303 FORMAT(1X,I4,' *** IMMEDIATE VALUE > 256 ***')                    ASM01080
      GO TO 35                                                          ASM01090
  204 WRITE(LU,304)LINE                                                 ASM01100
```

```
304 FORMAT(1X,I4,' *** ADDRESS OUTSIDE ADDRESS SPACE ***')                ASM01110
    GO TO 35                                                             ASM01120
205 WRITE(LU,305)LINE                                                    ASM01130
305 FORMAT(1X,I4,' *** INVALID INDIRECT ADDRESSING ***')                 ASM01140
    GO TO 35                                                             ASM01150
206 WRITE(LU,306)LINE                                                    ASM01160
306 FORMAT(1X,I4,' *** INVALID RELATIVE ADDRESS ***')                    ASM01170
    GO TO 35                                                             ASM01180
207 WRITE(LU,307)LINE                                                    ASM01190
307 FORMAT(1X,I4,' *** ILLEGAL ADDRESSING MODE ***')                     ASM01200
    GO TO 35                                                             ASM01210
208 WRITE(LU,308)LINE                                                    ASM01220
308 FORMAT(1X,I4,' *** INVALID OP CODE ***')                             ASM01230
    GO TO 35                                                             ASM01240
209 WRITE(LU,309)LINE                                                    ASM01250
309 FORMAT(1X,I4,' *** CONSTANT TOO LARGE ***')                          ASM01260
    GO TO 35                                                             ASM01270
210 WRITE(LU,310)LINE                                                    ASM01280
310 FORMAT(1X,I4,' *** MORE THAN 40 ASCII CHARACTERS ***')               ASM01290
    GO TO 35                                                             ASM01300
211 WRITE(LU,311)LINE                                                    ASM01310
311 FORMAT(' ')                                                          ASM01320
 35 IF(PASS.GT.1)GO TO 40                                                ASM01330
    IF(JK.GT.1)PC=PC+BYTE*JK-BYTE                                        ASM01340
    PC=PC+BYTE                                                           ASM01350
    GO TO 10                                                             ASM01360
C                                                                        ASM01370
C   PUNCH OUTPUT TAPE AND LISTING                                        ASM01380
C                                                                        ASM01390
 40 IF(PASS.NE.IPASS)GO TO 100                                           ASM01400
    IF(JK.GT.0)GO TO 60                                                  ASM01410
    CALL CONV1(IPC,TWO,IPTR,MONE,PC)                                     ASM01420
    PC=PC+BYTE                                                           ASM01430
    IF(BYTE.EQ.0.OR.BYTE.GT.3)GO TO 55                                   ASM01440
    CALL CONV1(IOPCD,ONE,IPTR,MONE,CODE(1))                              ASM01450
    WRITE(LU,3)LINE,IPC(2),IPC(1),IOPCD,IN                               ASM01460
  3 FORMAT(1X,I4,2X,2A2,1X,A2,2X,40A2)                                   ASM01470
    CALL CONV1(IADD,TWO,IPTR,MONE,CODE(2))                               ASM01480
    IF(MN.NE.0)GO TO 59                                                  ASM01490
    POUT(1)=IPC(2)                                                       ASM01500
    POUT(2)=IPC(1)                                                       ASM01510
    MN=2                                                                 ASM01520
 59 POUT(MN+1)=IOPCD                                                     ASM01530
    POUT(MN+2)=IADD(1)                                                   ASM01540
    POUT(MN+3)=IADD(2)                                                   ASM01550
    MN=MN+BYTE                                                           ASM01560
    IF(MN.GE.MNMAX)CALL PUNCH(POUT,MN,LUP)                               ASM01570
    IF(BYTE.EQ.1)GO TO 10                                                ASM01580
    BYTE=BYTE-1                                                          ASM01590
    WRITE(LU,4)(IADD(I),I=1,BYTE)                                        ASM01600
  4 FORMAT(12X,A2)                                                       ASM01610
    GO TO 10                                                             ASM01620
 55 WRITE(LU,7)LINE,IPC(2),IPC(1),IN                                     ASM01630
  7 FORMAT(1X,I4,2X,2A2,5X,40A2)                                         ASM01640
    IF(JK.LT.0.OR.BYTE.GT.3)GO TO 56                                     ASM01650
```

```
      GO TO 10                                                          ASM01660
   56 IF(MN.GT.2)CALL PUNCH(POUT,MN,LUP)                                ASM01670
      GO TO 10                                                          ASM01680
   60 DO 61 KK=1,JK                                                     ASM01690
      CALL CONV1(IPC,TWO,IPTR,MONE,PC)                                  ASM01700
      IF(MN.NE.0)GO TO 63                                               ASM01710
      POUT(1)=IPC(2)                                                    ASM01720
      POUT(2)=IPC(1)                                                    ASM01730
      MN=2                                                              ASM01740
   63 PC=PC+BYTE                                                        ASM01750
      CODE(1)=OBUF(KK)                                                  ASM01760
      CALL CONV1(IADD,BYTE,IPTR,MONE,CODE(1))                           ASM01770
      POUT(MN+1)=IADD(1)                                                ASM01780
      POUT(MN+2)=IADD(2)                                                ASM01790
      MN=MN+BYTE                                                        ASM01800
      IF(MN.GE.MNMAX)CALL PUNCH(POUT,MN,LUP)                            ASM01810
      IF(KK.EQ.1)GO TO 62                                               ASM01820
      WRITE(LU,9)IPC(2),IPC(1),IADD(1)                                  ASM01830
    9 FORMAT(7X,2A2,1X,A2)                                              ASM01840
      IF(BYTE.EQ.2)WRITE(LU,4)IADD(2)                                   ASM01850
   61 CONTINUE                                                          ASM01860
      GO TO 10                                                          ASM01870
   62 WRITE(LU,3)LINE,IPC(2),IPC(1),IADD(1),IN                          ASM01880
      IF(BYTE.EQ.2)WRITE(LU,4)IADD(2)                                   ASM01890
      GO TO 61                                                          ASM01900
  100 IF(PASS.EQ.3)WRITE(LU,2)LINE,IN                                   ASM01910
      J=PASS-1                                                          ASM01920
      IF(LU.NE.1)WRITE(LU,75)J,NERR                                     ASM01930
   75 FORMAT(/' END PASS ',I1,':',1X,I3,' ERRORS'/)                     ASM01940
      IF(PASS.EQ.3)GO TO 76                                             ASM01950
      LINE=0                                                            ASM01960
      NERR=0                                                            ASM01970
      IPASS=2                                                           ASM01980
      PC=0                                                              ASM01990
      REWIND DSK                                                        ASM02000
      DSW=1                                                             ASM02010
      GO TO 10                                                          ASM02020
   76 IF(MN.GT.0)CALL PUNCH(POUT,MN,LUP)                                ASM02030
      WRITE(LUP,13)                                                     ASM02040
   13 FORMAT(';00')                                                     ASM02050
      STOP                                                              ASM02060
      END                                                               ASM02070
```



```
      FUNCTION ISOLT(IPTR,IBUF)                                         ASM02080
      IMPLICIT INTEGER*2 (A-Z)                                          ASM02090
      INTEGER IPTR                                                      ASM02100
C     THIS FUNCTION ISOLATES THE IPTR'TH CHARACTER IN IBUF SO           ASM02110
C     THAT ISOLT CONTAINS THE CHARACTER AND A SPACE                     ASM02120
      DIMENSION IBUF(1)                                                 ASM02130
      DATA BLANK/' '/                                                   ASM02140
      IK=(IPTR+1)/2                                                     ASM02150
      IK1=IPTR+1-2*IK                                                   ASM02160
      IF(IK1.EQ.0)ISOLT=MASK2(SR8(IBUF(IK)))                            ASM02170
      IF(IK1.EQ.1)ISOLT=MASK2(IBUF(IK))                                 ASM02180
      RETURN                                                            ASM02190
      END                                                               ASM02200
```

```
      SUBROUTINE PROC(LABEL,LNTH,NSYT,SYMB,ISYMB,BYTE,PASS,VAL,          ASM02210
     * IERROR,MODE,PC)                                                  ASM02220
      IMPLICIT INTEGER*2 (A-Z)                                          ASM02230
C                                                                       ASM02240
C    THIS SUBROUTINE SORTS OUT THE VARIOUS ADDRESS MODES AND DETERMINES ASM02250
C    THE VALUE OF THE OPERAND                                           ASM02260
C                                                                       ASM02270
C    LABEL           OPERAND FIELD                                      ASM02280
C    LNTH            LENGTH OF OPERAND FIELD                            ASM02290
C    NSYT            NUMBER OF ENTRIES IN SYMBOL TABEL                  ASM02300
C    SYMB, ISYMB     SYMBOL TABLE                                       ASM02310
C    BYTE            LENGTH OF INSTRUCTION                              ASM02320
C    PASS            ASSEMBLER PASS NUMBER                              ASM02330
C    VAL             VALUE OF OPERAND                                   ASM02340
C    IERROR          ERROR FLAG                                         ASM02350
C    MODE            ADDRESSING MODE                                    ASM02360
C    PC              PROGRAM COUNTER                                    ASM02370
C                                                                       ASM02380
      DIMENSION LABEL(1),ISYMB(3,1),ICP(3)                              ASM02390
      DATA     AT/'@'/,DOLLAR/'$'/,PCENT/'%'/,PLUS/'+'/,MINUS/'-'/      ASM02400
      DATA     COMMA/','/,BLANK/' '/,EQUAL/'='/,PAREN/'('/,A/'A'/       ASM02410
      DATA          X/'X'/,Y/'Y'/,CPAREN/')'/,QUOTE/''''/               ASM02420
      REAL VAL,PC                                                       ASM02430
      DIMENSION SYMB(1)                                                 ASM02440
      ICHAR=ISCLT(1,LABEL)                                              ASM02450
      IF(ICHAR.EQ.EQUAL)GO TO 10                                        ASM02460
      IF(ICHAR.EQ.PAREN)GO TO 20                                        ASM02470
      IF(LABEL(1).EQ.A)GO TO 30                                         ASM02480
C                                                                       ASM02490
C    ABS OR Z.PAGE ADDRESSING?                                          ASM02500
      IPTR=0                                                            ASM02510
      CALL LABPR(LABEL,LNTH,IPTR,VAL,NSYT,SYMB,ISYMB,IERROR,PC)         ASM02520
      IF(VAL.GT.65535.)IERROR=4                                         ASM02530
   47 IF(VAL.GE.256)BYTE=3                                              ASM02540
      IF(VAL.LT.256)BYTE=2                                              ASM02550
      IF(IERROR.NE.0)BYTE=3                                             ASM02560
      IF(PASS.EQ.1)RETURN                                               ASM02570
      IF(IPTR.EQ.LNTH)GO TO 50                                          ASM02580
      IF((IPTR+2).NE.LNTH)IERROR=2                                      ASM02590
      IF(ISCLT(IPTR+2,LABEL).EQ.X)GO TO 51                              ASM02600
      IF(ISCLT(IPTR+2,LABEL).EQ.Y)GO TO 52                              ASM02610
      IERROR=2                                                          ASM02620
C                                                                       ASM02630
C    INDEX ADDRESS MODES                                                ASM02640
C                                                                       ASM02650
   51 MODE=8                                                            ASM02660
      IF(VAL.LT.256.)MODE=7                                             ASM02670
      RETURN                                                            ASM02680
   52 MODE=9                                                            ASM02690
      IF(VAL.LT.256.)MODE=11                                            ASM02700
      RETURN                                                            ASM02710
C                                                                       ASM02720
C    IMMEDIATE ADDRESSING                                               ASM02730
   10 BYTE=2                                                            ASM02740
      IF(PASS.EQ.1)RETURN                                               ASM02750
```

```
      IPTR=1                                                                ASM02760
      ICHAR=ISCLT(IPTR+1,LABEL)                                             ASM02770
      IF(ICHAR.EQ.QUOTE)GO TO 12                                            ASM02780
      CALL LABPR(LABEL,LNTH,IPTR,VAL,NSYT,SYMB,ISYMB,IERROR,PC)             ASM02790
      IF(VAL.GT.255.)IERROR=3                                               ASM02800
      MODE=1                                                                ASM02810
      RETURN                                                                ASM02820
   12 VAL=SAB(LABEL(2))                                                     ASM02830
      IF(LNTH.EQ.2)VAL=32                                                   ASM02840
      IF(LNTH.GT.4)IERROR=2                                                 ASM02850
      MODE=1                                                                ASM02860
      RETURN                                                                ASM02870
C                                                                           ASM02880
C   ABSOLUTE AND ZERO PAGE ADDRESSING                                       ASM02890
C                                                                           ASM02900
   50 MODE=2                                                                ASM02910
      IF(VAL.LT.256)MODE=3                                                  ASM02920
      RETURN                                                                ASM02930
C                                                                           ASM02940
C   INDIRECT ADDRESSING                                                     ASM02950
C                                                                           ASM02960
   20 BYTE=2                                                                ASM02970
      IF(PASS.EQ.1)RETURN                                                   ASM02980
      IPTR=1                                                                ASM02990
      CALL LABPR(LABEL,LNTH,IPTR,VAL,NSYT,SYMB,ISYMB,IERROR,PC)             ASM03000
      IF(VAL.GT.65535.)IERROR=4                                             ASM03010
      IF(IPTR.EQ.LNTH)GO TO 21                                              ASM03020
      ICHAR=ISCLT(IPTR+1,LABEL)                                             ASM03030
      IF(ICHAR.EQ.CPAREN)GO TO 22                                           ASM03040
      IF(ISCLT(IPTR+2,LABEL).NE.X.OR.(IPTR+3).NE.LNTH)IERROR=2              ASM03050
C                                                                           ASM03060
C   INDEXED INDIRECT                                                        ASM03070
C                                                                           ASM03080
      IF(VAL.LT.0.OR.VAL.GT.256.)IERROR=5                                   ASM03090
      MODE=5                                                                ASM03100
      RETURN                                                                ASM03110
C                                                                           ASM03120
C   INDIRECT (JUMP ONLY)                                                    ASM03130
C                                                                           ASM03140
   21 MODE=10                                                               ASM03150
      RETURN                                                                ASM03160
   22 IF(ISCLT((IPTR+3),LABEL).NE.Y.OR.(IPTR+3).NE.LNTH)IERROR=2            ASM03170
C                                                                           ASM03180
C   INDIRECT INDEXED                                                        ASM03190
C                                                                           ASM03200
      IF(VAL.LT.0.OR.VAL.GT.256.)IERROR=5                                   ASM03210
      MODE=6                                                                ASM03220
      RETURN                                                                ASM03230
C                                                                           ASM03240
C   ACCUMULATOR ADDRESSING                                                  ASM03250
C                                                                           ASM03260
   30 BYTE=1                                                                ASM03270
      MODE=4                                                                ASM03280
      RETURN                                                                ASM03290
      END                                                                   ASM03300
```

```
      SUBROUTINE LABPR(LABEL,LNTH,IPTR,VAL,NSYT,SYMB,ISYMB,IERROR,PC)    ASM03310
      IMPLICIT INTEGER*2 (A-Z)                                           ASM03320
C                                                                        ASM03330
C   THIS ROUTINE DETERMINES THE VALUE OF A LABEL IN THE OPERAND ALONG    ASM03340
C   WITH AN ADDED OR SUBTRACTED CONSTANT                                 ASM03350
C                                                                        ASM03360
C   LABEL              LABEL FIELD                                       ASM03370
C   LNTH               LENGTH OF LABEL FIELD                             ASM03380
C   IPTR               POINTER IN FIELD                                  ASM03390
C   VAL                RESULTING VALUE OF LABEL                          ASM03400
C   NSYT               NUMBER OF SYMBOL TABLE ENTRIES                    ASM03410
C   SYMB,ISYMB         SYMBOL TABLE                                      ASM03420
C   IERROR             ERROR FLAG                                        ASM03430
C   PC                 PROGRAM COUNTER                                   ASM03440
C                                                                        ASM03450
      REAL ANSER,VAL,PC                                                  ASM03460
      INTEGER*2 ABET(26)/'A','B','C','D','E','F','G','H','I','J','K',   ASM03470
     * 'L','M','N','O','P','Q','R','S','T','U','V','W',                 ASM03480
     *'X','Y','Z'/                                                      ASM03490
      DATA    COMMA/','/,CPAREN/')'/,ZERO/'0'/,NINE/'9'/,BLANK/' '/,    ASM03500
     *  A/'A'/,Z/'Z'/,AT/'@'/,DOLLAR/'$'/,PLUS/'+'/,MINUS/'-'/,STAR/'*'/ASM03510
      DATA PCENT/'%'/                                                    ASM03520
      DIMENSION LABEL(1),SYMB(1),ISYMB(3,1),IOP(3)                       ASM03530
      REAL EASE                                                          ASM03540
      INTEGER KPTR                                                       ASM03550
      ONE=1                                                              ASM03560
C                                                                        ASM03570
C   LOOK FOR SYMBOL IN TABLE                                             ASM03580
C                                                                        ASM03590
      VAL=0                                                              ASM03600
      IADD=1                                                             ASM03610
      ICHAR=ISCLT(IPTR+1,LABEL)                                          ASM03620
      IF(ICHAR.EQ.STAR)GO TO 44                                          ASM03630
      DO 10 I=1,26                                                       ASM03640
      IF(ICHAR.EQ.ABET(I))GO TO 14                                       ASM03650
   10 CONTINUE                                                           ASM03660
      GO TO 13                                                           ASM03670
   14 IOP(1)=BLANK                                                       ASM03680
      IOP(2)=BLANK                                                       ASM03690
      IOP(3)=BLANK                                                       ASM03700
      LTH=6                                                              ASM03710
      CALL CWORD(LABEL,IPTR,LNTH,IOP,LTH,IFLAG,ONE)                      ASM03720
      IF(IFLAG.EQ.-1)GO TO 40                                            ASM03730
      IF(NSYT.EQ.0)GO TO 145                                             ASM03740
      DO 45 I=1,NSYT                                                     ASM03750
      IERR=IAB (IOP(1)-ISYMB(1,I))+IAB (IOP(2)-ISYMB(2,I))+              ASM03760
     *  IAB (IOP(3)-ISYMB(3,I))                                          ASM03770
      IF(IERR.EQ.0)GO TO 46                                              ASM03780
   45 CONTINUE                                                           ASM03790
C                                                                        ASM03800
C   LABEL NOT FOUND - ASSUME NOT BASE PAGE                               ASM03810
C                                                                        ASM03820
  145 IERROR=1                                                           ASM03830
      RETURN                                                             ASM03840
C                                                                        ASM03850
```

```
C    LABEL IS "*" - GET PC                                              ASM03860
C                                                                       ASM03870
   44 VAL=PC                                                            ASM03880
      IPTR=IPTR+1                                                       ASM03890
      GO TO 48                                                          ASM03900
   46 VAL=SYMB(I)                                                       ASM03910
   48 IF(IPTR.EQ.LNTH)GO TO 47                                          ASM03920
C                                                                       ASM03930
C    MORE OPERAND TO PROCESS                                            ASM03940
C                                                                       ASM03950
      KPTR=IPTR                                                         ASM03960
      ICHAR=ISOLT(KPTR,LABEL)                                           ASM03970
      IF(ICHAR.EQ.COMMA.OR.ICHAR.EQ.CPAREN)GO TO 49                     ASM03980
      IF(ICHAR.EQ.PLUS)IADD=1                                           ASM03990
      IF(ICHAR.EQ.MINUS)IADD=-1                                         ASM04000
   13 IPTR=IPTR+1                                                       ASM04010
      KPTR=IPTR                                                         ASM04020
      ICHAR=ISOLT(KPTR,LABEL)                                           ASM04030
      BASE=0                                                            ASM04040
      IF(ICHAR.EQ.AT)BASE=8.                                            ASM04050
      IF(ICHAR.EQ.DOLLAR)BASE=16.                                       ASM04060
      IF(ICHAR.EQ.PCENT)BASE=2.                                         ASM04070
      IF(ICHAR.GE.ZERO.AND.ICHAR.LE.NINE)BASE=10.                       ASM04080
      IF(BASE.EQ.0)GO TO 40                                             ASM04090
      IF(BASE.EQ.10.)IPTR=IPTR-1                                        ASM04100
      CALL CONV1(LABEL,LNTH,IPTR,BASE,ANSER)                            ASM04110
      IF(IADD.EQ.1)VAL=VAL+ANSER                                        ASM04120
      IF(IADD.EQ.-1)VAL=VAL-ANSER                                       ASM04130
   47 RETURN                                                            ASM04140
   49 IPTR=IPTR-1                                                       ASM04150
      RETURN                                                            ASM04160
   40 VAL=0                                                             ASM04170
      IERROR=2                                                          ASM04180
      RETURN                                                            ASM04190
      END                                                               ASM04200
```

```
      SUBROUTINE GWORD(IBUF,IBUFP,IBUFL,IPBF,IPBL,IFLAG,IFST)            ASM04210
      IMPLICIT INTEGER*2 (A-Z)                                           ASM04220
C   THIS SUBROUTINE TRANSFERS WORDS FROM ONE BUFFER TO ANOTHER. A        ASM04230
C   WORD IS DEFINED BY A TRAILING SPACE. INITIAL BLANKS ARE THROWN       ASM04240
C   OUT. IF AN ODD NUMBER OF BYTES ARE ENCOUNTERED, THE FINAL BYTE OF    ASM04250
C   THE OUTPUT BUFFER IS FILLED WITH A BLANK.                            ASM04260
C                                                                        ASM04270
C   IBUF      INPUT BUFFER IBUFL BYTES LONG                              ASM04280
C   IBUFP     POINTS TO LAST BYTE PROCESSED IN IBUF                      ASM04290
C   IPBF      OUTPUT BUFFER                                              ASM04300
C   IPBL      INPUT AS LENGTH (BYTES) OF IPBF, ON RETURN # BYTES MOVED   ASM04310
C   IFLAG     ON RETURN: 0 NORMAL RETURN                                 ASM04320
C                       -1 END OF IPBF ENCOUNTERED                       ASM04330
C                        1 END OF IBUF ENCOUNTERED                       ASM04340
C   IFST      1 - RECORD MAY END WITH +,-, OR ,                          ASM04350
C             0 RECORD ENDS WITH BLANK                                   ASM04360
C                                                                        ASM04370
      DIMENSION IBUF(1),IPBF(1)                                          ASM04380
      DATA    SPACE/' '/,CPAREN/')'/,PLUS/'+'/,MINUS/'-'/,COMMA/','/     ASM04390
      IFLAG=0                                                            ASM04400
      IPBLI=0                                                            ASM04410
   20 IBUFP=IBUFP+1                                                      ASM04420
      IF(IBUFP.GT.IBUFL)GO TO 100                                        ASM04430
C                                                                        ASM04440
      IK=(IBUFP+1)/2                                                     ASM04450
      IK1=IBUFP+1-2*IK                                                   ASM04460
      IF(IK1.EQ.0)ICHAR=SR8(IBUF(IK))                                    ASM04470
      IF(IK1.EQ.1)ICHAR=MASK(IBUF(IK))                                   ASM04480
C   TEST FOR LEADING SPACES                                              ASM04490
      IF(ICHAR.EQ.SPACE/256.AND.IPBLI.EQ.0)GO TO 20                      ASM04500
C   TEST FOR TRAILING SPACE                                              ASM04510
      IF(ICHAR.EQ.SPACE/256)GO TO 200                                    ASM04520
      IF(IFST.EQ.0)GO TO 21                                              ASM04530
      IF(ICHAR.EQ.PLUS/256.OR.ICHAR.EQ.MINUS/256.OR.ICHAR.EQ.COMMA/256)  ASM04540
     * GO TO 200                                                         ASM04550
      IF(ICHAR.EQ.CPAREN/256)GO TO 200                                   ASM04560
C   PACK NON-SPACE BYTES                                                 ASM04570
   21 IPBLI=IPBLI+1                                                      ASM04580
      IF(IPBLI.GT.IPBL)GO TO 300                                         ASM04590
      IJ=(IPBLI+1)/2                                                     ASM04600
      IJ1=IPBLI+1-2*IJ                                                   ASM04610
      IF(IJ1.EQ.0)IPBF(IJ)=MASK2(ICHAR)                                  ASM04620
      IF(IJ1.EQ.1)IPBF(IJ)=SR8(IPBF(IJ))*256+ICHAR                       ASM04630
      GO TO 20                                                           ASM04640
C   END OF INPUT BUFFER RETURN                                           ASM04650
  100 IFLAG=1                                                            ASM04660
      IBUFP=IBUFP-1                                                      ASM04670
  200 IPBL=IPBLI                                                         ASM04680
      RETURN                                                             ASM04690
C   OUTPUT BUFFER OVERFLOW                                               ASM04700
  300 IPBL=IPBLI-1                                                       ASM04710
      IFLAG=-1                                                           ASM04720
      RETURN                                                             ASM04730
      END                                                                ASM04740
```

```
      SUBROUTINE OPCD(IN,IBUFP,BYTE,CODE,PASS,NSYT,ISYMB,SYMB,PC,      ASM04750
     1IERROR,OBUF,JK)                                                  ASM04760
      IMPLICIT INTEGER*2 (A-Z)                                         ASM04770
C                                                                      ASM04780
C     THIS SUBROUTINE DECODES THE OPCODES AND CALLS PROC TO DECODE     ASM04790
C     LABELS. ON PASS 1 IT RETURNS WITH THE NUMBER OF BYTES REQUIRED   ASM04800
C     FOR THE INSTRUCTION. ON PASS 2 IT ALSO RETURNS THE CODE IN       ASM04810
C     ARRAY CODE.                                                      ASM04820
C                                                                      ASM04830
C                                                                      ASM04840
C     IN              INPUT BUFFER                                     ASM04850
C     IBUFP           INPUT BUFFER POINTER                             ASM04860
C     BYTE            NUMBER OF BYTES FOR INSTRUCTION                  ASM04870
C     CODE            REAL BUFFER CONTAINING THE CODE                  ASM04880
C     PASS            ASMB PASS                                        ASM04890
C     NSYT            NUMBER OF ENTRIES IN SYMBOL TABLE                ASM04900
C     ISYMB           SYMBOL TABLE                                     ASM04910
C     SYMB            SYMBOL TABLE ADRESSES                            ASM04920
C     PC              PROGRAM COUNTER                                  ASM04930
C     IERROR          ERROR FLAG                                       ASM04940
C     OBUF            CONSTANT DEFINITION BUFFER                       ASM04950
C     JK              NUMBER OF CONSTANTS IN OBUF OR -1 FOR ORG OR BSS ASM04960
C                                                                      ASM04970
C                                                                      ASM04980
      DIMENSION NOPC(2,70),IN(1),ISYMB(3,1),OBUF(1)                    ASM04990
      DIMENSION OPCF(3),LABEL(40),MIST(33),MIST2(11,22)                ASM05000
      REAL VAL,PC,CODE                                                 ASM05010
      DIMENSION CODE(1),SYMB(1)                                        ASM05020
      DATA PLUS/1H+/,MINUS/1H-/                                        ASM05030
      DATA NOPC/2HBR,2HK ,2HCL,2HC ,2HCL,2HD ,2HCL,2HI ,2HCL,2HV ,     ASM05040
     & 2HDE,2HX ,2HDE,2HY ,2HIN,1HX,2HIN,1HY,2HNO,1HP,2HPH,1HA,       ASM05050
     & 2HPH,1HP,2HPL,1HA,2HPL,1HP,2HPT,1HI,2HPT,1HS,2HSE,1HC,         ASM05060
     % 2HSE,1HD,2HSE,1HI,2HTA,1HX,2HTA,1HY,2HTS,1HX,2HTX,1HA,         ASM05070
     # 2HTX,1HS,2HTY,1HA,2HBC,1HC,2HBC,1HS,2HBE,1HQ,2HBM,1HI,         ASM05080
     $ 2HBN,1HE,2HBP,1HL,2HBV,1HC,2HBV,1HS,2HAD,1HC,2HAN,1HD,         ASM05090
     & 2HAS,1HL,2HBI,1HT,2HCM,1HP,2HCP,1HX,2HCP,1HY,2HDE,1HC,         ASM05100
     % 2HEO,1HR,2HIN,1HC,2HJM,1HP,2HJS,1HR,2HLD,1HA,2HLD,1HX,2HLD,1HY,ASM05110
     $ 2HLS,1HR,2HOR,1HA,2HPO,1HL,2HSB,1HC,2HST,1HA,2HST,1HX,2HST,1HY,ASM05120
     # 8*2H  ,2HOR,1HG,2HEN,1HD,2HEQ,1HU,2HBS,1HS,2HAD,1HR,2HAS,1HC ,  ASM05130
     " 2HOC,1HT,2HRE,1HX,2HDC,1HM,2HIN,1HT,2HBC,1HD/                   ASM05140
      DATA MIST/0,24,216,88,184,202,136,232,200,234,72,8,104,40,       ASM05150
     & 64,96,56,248,120,170,168,186,138,154,152,144,176,240,48,       ASM05160
     % 208,16,80,112/                                                 ASM05170
      DATA MIST2/105,109,101,-1,97,113,117,125,121,-1,-121,            ASM05180
     & 41,45,37,-1,33,49,53,61,57,-1,-57,-1,14,6,10,-1,-1,22,30,3*-1,  ASM05190
     % -1,44,36,8*-1,201,205,197,-1,193,209,213,221,217,-1,-217,       ASM05200
     $ 224,236,228,8*-1, 192,204,196,8*-1,                            ASM05210
     # -1,206,198,3*-1,214,222,3*-1,73,77,69,-1,65,81,85,93,89,-1,-89, ASM05220
     " -1,238,238,3*-1,246,254,3*-1, -1,76,76,6*-1,108,-1,            ASM05230
     % -1,2*32,8*-1,169,173,165,-1,161,177,181,189,185,-1,-185,       ASM05240
     $ 162,174,166,5*-1,190,-1,182,160,172,164,3*-1,180,188,3*-1,     ASM05250
     # -1,78,70,74,2*-1,86,94,3*-1, 9,13,5,-1,1,17,21,29,25,-1,-25,   ASM05260
     " -1,46,38,42,2*-1,54,62,3*-1,                                   ASM05270
     & 233,237,229,-1,225,241,245,253,249,-1,-249,                    ASM05280
     " -1,141,133,-1,129,145,149,157,153,-1,-153, -1,142,134,7*-1,150, ASM05290
     % -1,140,132,3*-1,148,4*-1/                                      ASM05300
C                                                                      ASM05310
C     FIND OPCODE                                                      ASM05320
C                                                                      ASM05330
      ZERO=0                                                           ASM05340
      ALTY=80                                                          ASM05350
      JK=0                                                             ASM05360
```

```
      LOPCD=4                                                          ASM05370
      CALL GWORD(IN,IBUFP,AITY,OPCF,LOPCD,IFLAG,ZRO)                   ASM05380
      IF(IFLAG.NE.0)IERROR=8                                           ASM05390
      DO 10 I=1,70                                                     ASM05400
      IERR=IAB (OPCF(1)-NOPC(1,I))+IAB (OPCF(2)-NOPC(2,I))             ASM05410
      IF(IERR.EQ.0)GO TO 11                                            ASM05420
10    CONTINUE                                                         ASM05430
      IERROR=8                                                         ASM05440
      CODE(1)=-1                                                       ASM05450
      BYTE=0                                                           ASM05460
      RETURN                                                           ASM05470
11    IF(I.GT.33)GO TO 50                                              ASM05480
C                                                                      ASM05490
C     OPCODES WITH IMPLIED OR RELATIVE ADDRESSING                      ASM05500
C                                                                      ASM05510
      IF(I.LT.26)BYTE=1                                                ASM05520
      IF(I.GT.25)BYTE=2                                                ASM05530
      IF(PASS.EQ.1)RETURN                                              ASM05540
      CODE(1)=MIST(I)                                                  ASM05550
      IF(I.LT.26)RETURN                                                ASM05560
      LABL=80                                                          ASM05570
      CALL GWORD(IN,IBUFP,AITY,LABEL,LABL,IFLAG,ZRO)                   ASM05580
      IF(IFLAG.NE.0)GO TO 110                                          ASM05590
      ICHAR=ISCLT(1,LABEL)                                             ASM05600
      IPTR=1                                                           ASM05610
      IADD=0                                                           ASM05620
      IF(ICHAR.EQ.PLUS)IADD=1                                          ASM05630
      IF(ICHAR.EQ.MINUS)IADD=-1                                        ASM05640
      IF(IADD.EQ.0)IPTR=0                                              ASM05650
      CALL LABPR(LABEL,LABL,IPTR,VAL,NSYT,SYMB,ISYMB,IERROR,PC)        ASM05660
      IF(IADD.EQ.1)CODE(2)=VAL                                         ASM05670
      IF(IADD.EQ.-1)CODE(2)=-VAL                                       ASM05680
      IF(IADD.EQ.0)CODE(2)=VAL-(PC+2)                                  ASM05690
      IF(CODE(2).LT.-128.OR.CODE(2).GT.128)IERROR=6                    ASM05700
      RETURN                                                           ASM05710
C                                                                      ASM05720
50    LABL=80                                                          ASM05730
      IF(I.EQ.61)GO TO 60                                              ASM05740
      IF(I.GT.64.AND.I.LT.71)GO TO 70                                  ASM05750
C                                                                      ASM05760
C     OPCODES WITH MULTIPLE ADDRESSING MODES                           ASM05770
C                                                                      ASM05780
      CALL GWORD(IN,IBUFP,AITY,LABEL,LABL,IFLAG,ZRO)                   ASM05790
      IF(IFLAG.NE.0)GO TO 110                                          ASM05800
      CALL PROC(LABEL,LABL,NSYT,SYMB,ISYMB,BYTE,PASS,VAL,              ASM05810
     & IERROR,MODE,PC)                                                 ASM05820
      IF(I.LT.59)GO TO 52                                              ASM05830
C                                                                      ASM05840
C     ORG, EQU, BSS, AND ADR PSUEDO INSTRUCTIONS                       ASM05850
C                                                                      ASM05860
C                                                                      ASM05870
      BYTE=0                                                           ASM05880
      IF(I.EQ.60)PC=VAL                                                ASM05890
      IF(I.EQ.60)JK=-1                                                 ASM05900
      IF(I.EQ.62.AND.PASS.EQ.1)SYMB(NSYT)=VAL                          ASM05910
      CODE(1)=0                                                        ASM05920
      CODE(2)=0                                                        ASM05930
      IF(I.EQ.63)BYTE=VAL                                              ASM05940
      IF(I.NE.64)RETURN                                                ASM05950
      BYTE=2                                                           ASM05960
      CODE(1)=VAL                                                      ASM05970
      CODE(2)=VAL/256                                                  ASM05980
      IF(VAL.LT.256.AND.PC.LT.256)BYTE=1                               ASM05990
      RETURN                                                           ASM06000
```

```
C                                                                       ASM06010
C       MULTIPLE MODE INSTRUCTION STORE                                 ASM06020
C                                                                       ASM06030
52      IF(I.EQ.44)BYTE=3                                               ASM06040
        IF(I.EQ.45)BYTE=3                                               ASM06050
        IF(PASS.EQ.1)RETURN                                             ASM06060
        CODE(1)=MIST2(MODE,I-33)                                        ASM06070
        IF(CODE(1).EQ.-1)GO TO 208                                      ASM06080
        IF (CODE(1).GT.0)GO TO 207                                      ASM06090
        CODE(1)=-CODE(1)                                                ASM06100
        BYTE=3                                                          ASM06110
        GO TO 207                                                       ASM06120
  208   IERROR=7                                                        ASM06130
  207   CODE(2)=VAL                                                     ASM06140
        RETURN                                                          ASM06150
60      PASS=PASS+1                                                     ASM06160
        RETURN                                                          ASM06170
C                                                                       ASM06180
C       PROCESS REMAINING PSUEDO INSTRUCTIONS                           ASM06190
C                                                                       ASM06200
70      CALL NUMBR(IN,IBUFP,OBUF,JK,I,BYTE,IERROR)                      ASM06210
        RETURN                                                          ASM06220
110     IERROR=2                                                        ASM06230
        RETURN                                                          ASM06240
        END                                                             ASM06250
```

```
      SUBROUTINE SYMBT(LABEL,PC,ISYMB,NSYTM,N,NSYT,SYMB,LINE,NERR)      ASM06260
      IMPLICIT INTEGER*2 (A-Z)                                          ASM06270
C                                                                       ASM06280
C   THIS PROGRAM CHECKS LABELS AND ENTERS VALID LABELS INTO SYMBOL      ASM06290
C   TABLE IN PASS 1.                                                    ASM06300
C                                                                       ASM06310
C   LABEL     LABEL TO BE ENTERED (LABEL(3))                            ASM06320
C   PC        PROGRAM COUNTER                                           ASM06330
C   ISYMB     SYMBOL TABLE (ISYMB(3,NSYTM))                             ASM06340
C   NSYTM     MAX LENGTH OF SYMBOL TABLE                                ASM06350
C   N         L. U. FOR ERROR OUTPUT                                    ASM06360
C   NSYT      LOC OF LAST TABLE ENTRY                                   ASM06370
C   SYMB      SYMBOL TABLE ADDRESSES                                    ASM06380
C   LINE      SOURCE LINE COUNTER                                       ASM06390
C   NERR      # OF ERRORS IN PASS                                       ASM06400
C                                                                       ASM06410
      DATA SPACE/' '/                                                   ASM06420
      INTEGER*2 ABET(26)/'A','B','C','D','E','F','G','H','I','J','K',  ASM06430
     * 'L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/     ASM06440
      REAL PC                                                           ASM06450
      INTEGER N                                                         ASM06460
      DATA    A/'A'/,X/'X'/,Y/'Y'/,S/'S'/,P/'P'/                        ASM06470
      DIMENSION LABEL(3),ISYMB(3,1),SYMB(1)                             ASM06480
      LL=LABEL(1)                                                       ASM06490
      IF(LL.EQ.A.OR.LL.EQ.X.OR.LL.EQ.Y.OR.LL.EQ.S.OR.LL.EQ.P)GO TO 40   ASM06500
      ICHAR=MASK2(SR8(LABEL(1)))                                        ASM06510
      DO 15 J=1,26                                                      ASM06520
      IF(ICHAR.EQ.ABET(J))GO TO 100                                     ASM06530
   15 CONTINUE                                                          ASM06540
      WRITE(N,1)LINE,LABEL                                              ASM06550
    1 FORMAT(I5,4H *** ,3A2,' INVALID. MUST START WITH LETTER***')      ASM06560
      NERR=NERR+1                                                       ASM06570
      RETURN                                                            ASM06580
C                                                                       ASM06590
C   SEARCH SYMBOL TABLE FOR DUPLICATE ENTRIES                           ASM06600
C                                                                       ASM06610
  100 IF(NSYT.EQ.0)GO TO 200                                            ASM06620
      DO 10 I=1,NSYT                                                    ASM06630
      ERR=IAB (LABEL(1)-ISYMB(1,I))+IAB (LABEL(2)-ISYMB(2,I))+          ASM06640
     * IAB (LABEL(3)-ISYMB(3,I))                                        ASM06650
      IF(ERR.EQ.0)GO TO 20                                              ASM06660
   10 CONTINUE                                                          ASM06670
  200 NSYT=NSYT+1                                                       ASM06680
      IF(NSYT.GT.NSYTM)GO TO 30                                         ASM06690
C   ENTER LABEL                                                         ASM06700
      ISYMB(1,NSYT)=LABEL(1)                                            ASM06710
      ISYMB(2,NSYT)=LABEL(2)                                            ASM06720
      ISYMB(3,NSYT)=LABEL(3)                                            ASM06730
      SYMB(NSYT)=PC                                                     ASM06740
      RETURN                                                            ASM06750
   20 WRITE(N,2)LINE,LABEL                                              ASM06760
    2 FORMAT(I5,' *** ',3A2,' IS DUPLICATE LABEL ***')                  ASM06770
      NERR=NERR+1                                                       ASM06780
      RETURN                                                            ASM06790
   30 WRITE(N,3)                                                        ASM06800
    3 FORMAT(' *** SYMBOL TABLE OVERFLOW ***')                          ASM06810
      STOP                                                              ASM06820
   40 WRITE(N,5)LINE,LABEL(1)                                           ASM06830
    5 FORMAT(I5,' *** ',1A2,' IS A RESERVED SYMBOL ***')                ASM06840
      NERR=NERR+1                                                       ASM06850
      RETURN                                                            ASM06860
      END                                                               ASM06870
```

```
      SUBROUTINE NUMBR(IN,IPTR,OBUF,JK,I,BYTE,IERROR)                    ASM06880
      IMPLICIT INTEGER*2 (A-Z)                                           ASM06890
C                                                                        ASM06900
C   THIS SUBROUTINE DECODES OPERAND OF PSEUDO OPS ASC,OCT,HEX,DCM,       ASM06910
C   INT, AND BCD.                                                        ASM06920
C                                                                        ASM06930
C   IN         INPUT BUFFER                                            ASM06940
C   IPTR       POINTER LOCATION FOR IN                                 ASM06950
C   OBUF       OUTPUT BUFFER FOR CONSTANTS                             ASM06960
C   JK         NUMBER OF CONSTANTS IN OBUF                             ASM06970
C   I          OPCODE NUMBER                                           ASM06980
C   BYTE       NUMBER OF BYTES IN CONSTANT                             ASM06990
C   IERROR     ERROR FLAG                                              ASM07000
      REAL ANSER                                                         ASM07010
      DATA     PLUS/'+'/,MINUS/'-'/,QUOTE/''''/,COMMA/','/              ASM07020
      DIMENSION OBUF(1),IN(1),NUM(3)                                     ASM07030
      INTEGER*2 ASCI(64)/32,65,66,67,68,69,70,71,                        ASM07040
     * 72,73,Z5D,Z2E,Z3C,Z28,Z2B,Z5E,Z26,74,75,76,77,78,79,80,           ASM07050
     * 81,82,Z58,36,42,41,Z3B,Z21,Z2D,Z2F,83,84,85,86,87,88,             ASM07060
     * 89,Z5A,0,Z2C,Z25,Z2D,Z3E,Z3F,Z30,49,50,51,52,53,54,55,            ASM07070
     * 55,57,58,Z23,64,Z27,Z3D,Z22/                                      ASM07080
      REAL BASE                                                          ASM07090
      INTEGER KPTR                                                       ASM07100
      AITY=80                                                            ASM07110
      ZRO=0                                                              ASM07120
      ONE=1                                                              ASM07130
      BYTE=1                                                             ASM07140
      IF(I.EQ.66.OR.I.EQ.68)BYTE=2                                       ASM07150
      JK=0                                                               ASM07160
      JPTR=IPTR                                                          ASM07170
      LNTH=80                                                            ASM07180
      CALL GWORD(IN,JPTR,AITY,OBUF,LNTH,IFLAG,ZRO)                       ASM07190
      IPTR=JPTR-LNTH-1                                                   ASM07200
      III=I-64                                                           ASM07210
      GO TO (65,20,30,40,40,40),III                                      ASM07220
C                                                                        ASM07230
C   SET BASE                                                             ASM07240
C                                                                        ASM07250
   20 BASE=8                                                             ASM07260
      GO TO 10                                                           ASM07270
   30 BASE=16                                                            ASM07280
      GO TO 10                                                           ASM07290
   40 BASE=10                                                            ASM07300
   10 IM=0                                                               ASM07310
C  LOOP TO PROCESS OPERAND                                               ASM07320
   11 IPBL=6                                                             ASM07330
      CALL GWORD(IN,IPTR,AITY,NUM,IPBL,IFLAG,ONE)                        ASM07340
      KPTR=IPTR                                                          ASM07350
      ICHAR=ISCLT(KPTR,IN)                                               ASM07360
      IF(ICHAR.EQ.PLUS.OR.ICHAR.EQ.MINUS)GO TO 100                       ASM07370
   50 IBUFF=0                                                            ASM07380
    2 FORMAT(6A2)                                                        ASM07390
      CALL CONV1(NUM,IPBL,IBUFF,BASE,ANSER)                              ASM07400
      JK=JK+1                                                            ASM07410
      CH=ANSER/BYTE**8                                                   ASM07420
      IF(CH.GT.256)IERROR=5                                              ASM07430
      IF(I.EQ.70)GO TO 110                                               ASM07440
      IF(IM.EQ.1)ANSER=-ANSER                                            ASM07450
      IF(ANSER.GT.32767.)ANSER=ANSER-65536.                              ASM07460
```

```
      CBUF(JK)=ANSER                                                    ASM07470
    1 FORMAT(2I8)                                                       ASM07480
      IF(IPTR.NE.JPTR)GO TO 10                                          ASM07490
      BYTE=1                                                            ASM07500
      IF(I.EQ.66.OR.I.EQ.68)BYTE=2                                      ASM07510
      RETURN                                                            ASM07520
  100 IF(ICHAR.EQ.MINUS)IM=1                                            ASM07530
      GO TO 11                                                          ASM07540
C   ECD PROCESSOR                                                       ASM07550
  110 IF(IM.EQ.1)ANSER=100.-ANSER                                       ASM07560
      JND=ANSER/10.                                                     ASM07570
      CBUF(JK)=JND*16+ANSER-JND*10                                      ASM07580
      IF(IPTR.NE.JPTR)GO TO 10                                          ASM07590
      BYTE=1                                                            ASM07600
      RETURN                                                            ASM07610
C     ASCII PROCESSOR                                                   ASM07620
   65 IPTR=IPTR+1                                                       ASM07630
      BYTE=1                                                            ASM07640
   66 IPTR=IPTR+1                                                       ASM07650
      KPTR=IPTR                                                         ASM07660
      ICHAR=ISOLT(KPTR,IN)                                              ASM07670
      IF(ICHAR.EQ.QUOTE)RETURN                                          ASM07680
      JK=JK+1                                                           ASM07690
      IF(JK.GT.40)GO TO 67                                              ASM07700
      JCHAR=SRB(ICHAR)                                                  ASM07710
      KCHAR=JCHAR/64*64                                                 ASM07720
      JCHAR=JCHAR-KCHAR                                                 ASM07730
      CBUF(JK)=ASCI(JCHAR+1)                                            ASM07740
      GO TO 66                                                          ASM07750
   67 JK=JK-1                                                           ASM07760
      IERROR=10                                                         ASM07770
      END                                                               ASM07780
```

```
      SUBROUTINE CONV1(IBUF,IBUFL,IPTR,BASE,ANSER)                      ASM07790
      IMPLICIT INTEGER*2 (A-Z)                                          ASM07800
C                                                                       ASM07810
C   THIS SUBROUTINE CONVERTS EITHER AN ASCII BUFFER IBUFL BYTES         ASM07820
C  LONG TO A FLOATING POINT NUMBER, OR CONVERTS A FLOATING POINT        ASM07830
C  NUMBER TO HEX.                                                       ASM07840
C                                                                       ASM07850
C  IBUF   BASE>0 INPUT ASCII BUFFER (ANY LENGTH)                        ASM07860
C         BASE<0 OUTPUT ASCII BUFFER HEX (1 OR 2 WORDS LONG)            ASM07870
C                                                                       ASM07880
C  IBUFL  FOR BASE >0 NUMBER OF BYTES TO BE PROCESSED                   ASM07890
C         FOR BASE <0 NUMBER OF WORDS TO BE OUTPUT (1 OR 2 ONLY)        ASM07900
C                                                                       ASM07910
C  IPTR   BASE>0 POINTS TO LAST BYTE PROCESSED                          ASM07920
C         BASE<0 NO FUNCTION                                            ASM07930
C                                                                       ASM07940
C  BASE   BASE>0 BASE OF CONVERSION                                     ASM07950
C         BASE<0 INDICATES CONVERSION TO ASCII HEX                      ASM07960
C                                                                       ASM07970
C  ANSER  BASE>0 FLOATING POINT RESULT OF CONVERSION                    ASM07980
C         BASE<0 FLOATING POINT SOURCE FOR CONVERSION                   ASM07990
C                                                                       ASM08000
      DIMENSION NUM(16),IWB(16),IBUF(1)                                 ASM08010
      REAL BASE                                                         ASM08020
      REAL ANSER                                                        ASM08030
      INTEGER KPTR                                                      ASM08040
      INTEGER IJ                                                        ASM08050
      REAL RJ                                                           ASM08060
      DATA NUM/'0','1','2','3','4','5','6','7','8','9','A','B','C',    ASM08070
     * 'D','E','F'/                                                    ASM08080
      IF(BASE.LT.0)GO TO 50                                             ASM08090
      IPTR=IPTR+1                                                       ASM08100
      JK=0                                                              ASM08110
   10 KPTR=IPTR                                                         ASM08120
      ICHAR=ISOLT(KPTR,IBUF)                                            ASM08130
      ICHAR=ISOLT(KPTR,IBUF)                                            ASM08140
      DO 20 I=1,16                                                      ASM08150
      IF(NUM(I).EQ.ICHAR)GO TO 21                                       ASM08160
   20 CONTINUE                                                          ASM08170
C  NON-RECOGNIZED SYMBOL                                                ASM08180
      IPTR=IPTR-1                                                       ASM08190
      GO TO 30                                                          ASM08200
   21 JK=JK+1                                                           ASM08210
      IWB(JK)=I-1                                                       ASM08220
      IF(IPTR.EQ.IBUFL)GO TO 30                                         ASM08230
      IPTR=IPTR+1                                                       ASM08240
      GO TO 10                                                          ASM08250
C  CONVERT RESULT                                                       ASM08260
   30 ANSER=0.                                                          ASM08270
      IF(JK.EQ.0)RETURN                                                 ASM08280
      DO 31 L=1,JK                                                      ASM08290
   31 ANSER=ANSER+BASE**(JK-L)*IWB(L)                                   ASM08300
      RETURN                                                            ASM08310
   50 IF(ANSER.LT.0)ANSER=65536.+ANSER                                  ASM08320
      RJ=ANSER                                                          ASM08330
      I=1                                                               ASM08340
   51 IJ=AMOD(RJ,256.)                                                  ASM08350
      IK=MOD(IJ,16)                                                     ASM08360
      IL=IJ/16                                                          ASM08370
      IBUF(I)=SR8(NUM(IK+1))+SR8(NUM(IL+1))*256                         ASM08380
      IF(IBUFL.EQ.I)RETURN                                              ASM08390
      I=I+1                                                             ASM08400
      RJ=ANSER/256                                                      ASM08410
      GO TO 51                                                          ASM08420
      END                                                               ASM08430
```

```
      SUBROUTINE PUNCH(POUT,MN,LUP)                                    ASM08440
      IMPLICIT INTEGER*2 (A-Z)                                         ASM08450
C                                                                      ASM08460
C     THIS SUBROUTINE CALCULATES THE CHECKSUM AND PUNCHES THE RECORD   ASM08470
C                                                                      ASM08480
C     POUT    BUFFER CONTAINING INFORMATION TO BE PUNCHED               ASM08490
C     MN      NUMBER OF ENTRIES IN POUT (INCLUDING P. C.)              ASM08500
C     LUP     LOGICAL UNIT OF PUNCH                                    ASM08510
C                                                                      ASM08520
      REAL AMN                                                         ASM08530
      REAL CHKSM,ANS                                                   ASM08540
      INTEGER LUP                                                      ASM08550
      DIMENSION POUT(1),ICHK(2)                                        ASM08560
      REAL MONE                                                        ASM08570
      MONE=-1.                                                         ASM08580
      TWO=2                                                            ASM08590
      ONE=1                                                            ASM08600
      CHKSM=MN-2                                                       ASM08610
      DO 10 I=1,MN                                                     ASM08620
      IPTR=0                                                           ASM08630
      IP=POUT(I)                                                       ASM08640
      CALL CONV1(IP,TWO,IPTR,16.,ANS)                                  ASM08650
      CHKSM=AMOD(CHKSM+ANS,65536.)                                     ASM08660
   10 CONTINUE                                                         ASM08670
      AMN=MN-2                                                         ASM08680
      CALL CONV1(NUM,ONE,IPTR,MONE,AMN)                                ASM08690
      CALL CONV1(ICHK,TWO,IPTR,MONE,CHKSM)                             ASM08700
      WRITE(LUP,1)NUM,(POUT(I),I=1,MN),ICHK(2),ICHK(1)                 ASM08710
    1 FORMAT(1H;,40A2)                                                 ASM08720
      MN=0                                                             ASM08730
      RETURN                                                           ASM08740
      END                                                              ASM08750
```

```
      INTEGER FUNCTION IAB*2(IN)                                       ASM08760
C     HALFWORD ABSOLUTE VALUE ROUTINE                                  ASM08770
      INTEGER*2 IN                                                     ASM08780
      IF(IN.LT.0)IAB=-IN                                               ASM08790
      IF(IN.GE.0)IAB=IN                                                ASM08800
      RETURN                                                           ASM08810
      END                                                              ASM08820
```

```
      INTEGER FUNCTION MASK2*2 (IN)                                    ASM08830
C     2-BYTE INPUT NUMBER IS SHIFTED LEFT 8 BITS AND A BLANK           ASM08840
C     IS ADDED IN THE SECOND BYTE.                                     ASM08850
      INTEGER*2 J(2),IN                                                ASM08860
      EQUIVALENCE(J(1),K)                                              ASM08870
      K=0                                                              ASM08880
      J(2)=IN                                                          ASM08890
      K=K*256+64                                                       ASM08900
      MASK2=J(2)                                                       ASM08910
      RETURN                                                           ASM08920
      END                                                              ASM08930
```

```
      INTEGER FUNCTION SR8*2(IN)                                        ASM08940
      IMPLICIT INTEGER*2 (A-Z)                                          ASM08950
C     HALFWORD INPUT IS SHIFTED RIGHT 8 BITS. ZEROS APPEAR IN           ASM08960
C     LEFT 8 BITS.                                                      ASM08970
      INTEGER K                                                         ASM08980
      INTEGER*2 J(2)                                                    ASM08990
      EQUIVALENCE(J(1),K)                                               ASM09000
      K=0                                                               ASM09010
      J(2)=IN                                                           ASM09020
      K=K/256                                                           ASM09030
      SR8=J(2)                                                          ASM09040
      RETURN                                                            ASM09050
      END                                                               ASM09060


      INTEGER FUNCTION MASK*2(IN)                                       ASM09070
C     HALFWORD INPUT: ROUTINE ZEROS LEFT 8 BITS                         ASM09080
      INTEGER*2 J(2),IN                                                 ASM09090
      EQUIVALENCE(J(1),K)                                               ASM09100
      K=0                                                               ASM09110
      J(2)=IN                                                           ASM09120
      K=K*256                                                           ASM09130
      J(1)=0                                                            ASM09140
      K=K/256                                                           ASM09150
      MASK=J(2)                                                         ASM09160
      RETURN                                                            ASM09170
      END                                                               ASM09180


      SUBROUTINE LFMT(IN)                                               ASM09190
C     FORMATS INPUT LINE WITH LABEL IN COL. 1-5, OPCODE IN 8-10,        ASM09200
C     AND OPERAND AND COMMENT SEPARATED BY 2 SPACES. HELPS USER         ASM09210
C     BY ALLOWING FREE-FORM INPUT.                                      ASM09220
      LOGICAL*1 IN(80),K(2),RES(80)                                     ASM09230
      INTEGER*2 BLK/' '/                                                ASM09240
      INTEGER*2 L                                                       ASM09250
      INTEGER IBUF(20),BK/' '/                                          ASM09260
      EQUIVALENCE (IBUF(1),RES(1)),(K(1),L)                             ASM09270
      DO 10 I=1,20                                                      ASM09280
   10 IBUF(I)=BK                                                        ASM09290
      L=BLK                                                             ASM09300
      M=1                                                               ASM09310
      K(2)=IN(1)                                                        ASM09320
      IF(L.EQ.BLK)GO TO 12                                              ASM09330
      RES(1)=IN(1)                                                      ASM09340
      DO 11 I=2,72                                                      ASM09350
```

```
      M=I                                                          ASM09360
      K(2)=IN(I)                                                   ASM09370
      IF(L.EQ.BLK)GO TO 12                                         ASM09380
   11 RES(I)=IN(I)                                                 ASM09390
      GO TO 99                                                     ASM09400
   12 J=8                                                          ASM09410
      M=M+1                                                        ASM09420
      DO 13 I=M,72                                                 ASM09430
      N=I                                                          ASM09440
      K(2)=IN(I)                                                   ASM09450
      IF(L.NE.BLK)GO TO 14                                         ASM09460
   13 CONTINUE                                                     ASM09470
      GO TO 99                                                     ASM09480
   14 M=8                                                          ASM09490
      DO 15 I=N,72                                                 ASM09500
      NN=I                                                         ASM09510
      K(2)=IN(I)                                                   ASM09520
      IF(L.EQ.BLK)GO TO 16                                         ASM09530
      RES(M)=IN(I)                                                 ASM09540
      M=M+1                                                        ASM09550
      IF(M.GT.72)GO TO 99                                          ASM09560
   15 CONTINUE                                                     ASM09570
      GO TO 99                                                     ASM09580
   16 NN=NN+1                                                      ASM09590
      DO 17 I=NN,72                                                ASM09600
      N=I                                                          ASM09610
      K(2)=IN(I)                                                   ASM09620
      IF(L.NE.BLK)GO TO 18                                         ASM09630
   17 CONTINUE                                                     ASM09640
      GO TO 99                                                     ASM09650
   18 M=12                                                         ASM09660
      DO 19 I=N,72                                                 ASM09670
      NN=I                                                         ASM09680
      K(2)=IN(I)                                                   ASM09690
      IF(L.EQ.BLK)GO TO 20                                         ASM09700
      RES(M)=IN(I)                                                 ASM09710
      M=M+1                                                        ASM09720
      IF(M.GT.72)GO TO 99                                          ASM09730
   19 CONTINUE                                                     ASM09740
      GO TO 99                                                     ASM09750
   20 M=M+2                                                        ASM09760
      DO 21 I=NN,72                                                ASM09770
      N=I                                                          ASM09780
      K(2)=IN(I)                                                   ASM09790
      IF(L.NE.BLK)GO TO 22                                         ASM09800
   21 CONTINUE                                                     ASM09810
      GO TO 99                                                     ASM09820
   22 DO 23 I=N,72                                                 ASM09830
      IF(M.GT.72)GO TO 99                                          ASM09840
      RES(M)=IN(I)                                                 ASM09850
   23 M=M+1                                                        ASM09860
   99 DO 100 I=1,72                                                ASM09870
  100 IN(I)=RES(I)                                                 ASM09880
      RETURN                                                       ASM09890
      END                                                          ASM09900
```

APPENDIX B

FILE: PROFILE EXEC A

```
&CONTROL OFF
CP SPOOL RDR CLASS *
&IF &READFLAG EQ 'CONSOLE &GOTO -NOSTK
&READ ARGS
-NOSTK CP SPOOL CONS TO * NOTERM START
SET RDYMSG SMSG
CP TERMINAL LINES 132
CP DEFINE T3350 AS 193 CYL 1
&STACK YES
&STACK RWL
FORMAT 193 C
ACCESS 193 C
CP PURGE RDR CL T
CP SPOOL CONSOLE TERM STOP PURGE
&TYPE READY: 1-CYL C-DISK ONLINE
&1 &2 &3 &4 &5 &6 &7 &8
&EXIT
```

FILE: JASM EXEC A

```
&CONTROL OFF NOMSG
GLOBAL TXTLIB FORTLIB
ERASE JOLT PRINTOUT C
FILEDEF 6 TERMINAL
FILEDEF 4 DISK JOLT HEXCODE C
FILEDEF 11 DISK JOLT CLEAN C1 (RECFM F BLKSIZE 80 LRECL 80)
&IF &INDEX LT 3 &GOTO -NOARG
STATE &1 &2 &3
&IF &RETCODE NE 0 &GOTO -FILE
FILEDEF 5 DISK &1 &2 &3 (RECFM F LRECL 80)
&IF &INDEX EQ 3 &GOTO -LOAD
&IF &4 EQ NOLIST &GOTO -NOLIST
-LOAD LOADMOD JOLT
START
&EXIT
-NOARG &IF &INDEX EQ 0 &EXIT
&IF &1 NE TERMINAL &GOTO -MSG
FILEDEF 5 TERMINAL
&GOTO -LOAD
-MSG &TYPE COMPLETE FILE NAME NOT SUPPLIED
&EXIT
-FILE &TYPE FILE &1 &2 &3 NOT FOUND
&EXIT
-NOLIST FILEDEF 6 DISK JOLT PRINTOUT C
&GOTO -LOAD
```

```
FILE: JERRS     EXEC      A

&CONTROL OFF NOMSG
STATE JCLT PRINTOUT C
&IF &RETCODE NE 0 &GOTO -FILE
&BEGSTACK
VERIFY ON 6 25
LOCATE / ENC PASS/
LOCATE / ENC PASS/
VERIFY ON 1 96
ZONE 1 5
TOP
CHANGE /-/-/ * *
QUIT
&END
E JCLT PRINTOUT C
&EXIT
-FILE &TYPE JCLT PRINTOUT C NOT FOUND
&EXIT
```

```
FILE: MASM      EXEC      A

&CONTROL OFF NOMSG
GLOBAL TXTLIB FORTLIB
ERASE MAINT PRINTOUT C
FILEDEF 6 TERMINAL
FILEDEF 4 DISK MAINT HEXCODE C
FILEDEF 11 DISK MAINT CLEAN C
&IF &INDEX LT 3 &GOTO -NOARG
STATE &1 &2 &3
&IF &RETCODE NE 0 &GOTO -FILE
FILEDEF 5 DISK &1 &2 &3
&IF &INDEX EQ 3 &GOTO -LOAD
&IF &4 EQ NOLIST &GOTO -NOLIST
-LOAD LOADMOD MAINT
START
&EXIT
-NOARG &IF &INDEX EQ 0 &EXIT
&IF &1 NE TERMINAL &GOTO -MSG
FILEDEF 5 TERMINAL
&GOTO -LOAD
-MSG &TYPE COMPLETE FILE NAME NOT SUPPLIED
&EXIT
-FILE &TYPE FILE &1 &2 &3 NOT FOUND
&EXIT
-NOLIST FILEDEF 6 DISK MAINT PRINTOUT C
&GOTO -LOAD
```

FILE: MBUILD EXEC A

```
&CONTROL OFF NOMSG
GLOBAL TXTLIB FORTLIB
&STACK LIFO NOSOURCE NOMAP
EXEC RWLFORT ASM6502
LOAD ASM6502 (NOMAP)
GENMOD MAINT MODULE C
ERASE MAINT PRINTOUT C
FILEDEF 6 TERMINAL
FILEDEF 4 DISK MAINT HEXCODE C
FILEDEF 11 DISK MAINT CLEAN C
&IF &INDEX LT 3 &GOTO -NOARG
STATE &1 &2 &3
&IF &RETCODE NE 0 &GOTO -FILE
FILEDEF 5 DISK &1 &2 &3
&IF &INDEX EQ 3 &GOTO -LOAD
&IF &4 EQ NOLIST &GOTO -NOLIST
-LOAD LOADMOD MAINT
START
&EXIT
-NOARG &IF &INDEX EQ 0 &EXIT
&IF &1 NE TERMINAL &GOTO -MSG
FILEDEF 5 TERMINAL
&GOTO -LOAD
-MSG &TYPE COMPLETE FILE NAME NOT SUPPLIED
&EXIT
-FILE &TYPE FILE &1 &2 &3 NOT FOUND
&EXIT
-NOLIST FILEDEF 6 DISK MAINT PRINTOUT C
&GOTO -LOAD
```